Stefano Papetti

Sound modeling issues in interactive sonification

From basic contact events to synthesis and manipulation tools

March 15, 2010

Università degli Studi di Verona
Dipartimento di Informatica

Advisor:
prof. Federico Fontana

Series N°: **TD-05-10**

Università di Verona
Dipartimento di Informatica
Strada le Grazie 15, 37134 Verona
Italy

a Monica

Ringraziamenti

Se nei tre anni e mezzo appena trascorsi ho avuto la possibilità di fare ricerca, lo devo principalmente (e cronologicamente) a Federico Avanzini, mio relatore presso l'Università di Padova, Davide Rocchesso dell'Università IUAV di Venezia, che mi ha accolto all'Università di Verona nel 2006, e Federico Fontana, mio tutore durante il dottorato. A loro vanno la mia più profonda stima e gratitudine. Essi rappresentano per me una costante fonte di stimoli per la ricerca, e l'esempio di come questa vada affrontata con entusiasmo e passione.

Il gruppo di ricerca (inteso in senso allargato, per le continue dislocazioni dei suoi componenti) di cui ho il piacere e l'onore di fare parte, è una felice combinazione di persone disponibili, preparate e dai vasti interessi, nonché di cari amici. Credo sia una rara e grande fortuna quella di poter lavorare in un collettivo come il nostro, dove il rapporto umano e l'affiatamento sono base fondante, e di questo sono a tutti estremamente riconoscente.

A Stefania Serafin un ringraziamento speciale per l'accoglienza in Danimarca (mentre era in dolce attesa della piccola Emma), e per il supporto burocratico-logistico-finanziario durante il mio periodo presso la Aalborg University di Copenaghen (luglio-dicembre 2008).

Il lavoro contenuto in questa tesi è frutto di numerosi contributi di diversa origine ed estrazione, e a questo proposito desidero ringraziare – oltre alle persone già citate – in ordine rigorosamente alfabetico: Marco Civolani, Stefano Delle Monache (alias SteM), Antonio De Sena, Delphine Devallez, Carlo Drioli, Pietro Polotti e Stefano Zambon (alias SteZ).

Acknowledgments

If during the three and a half years just passed I had the opportunity to do research, I owe it mainly (and chronologically) to Federico Avanzini, my Master's thesis advisor at the University of Padua, Davide Rocchesso from the IUAV University of Venice, who welcomed me to the University of Verona in 2006, and Federico Fontana, my tutor during the PhD. They deserve my deepest respect and gratitude. To me they represent a constant source of inspiration for research, and an example of how this should be approached with enthusiasm and passion.

The research group (to be understood in a broader sense, due to the continual displacements of its components) of which I have the pleasure and honor to be part, is a happy combination of open, competent people who have wide interests, as well as of dear friends. I am sure it is a rare and great luck to have the opportunity to work in a collective like ours, where human relations and teamwork are fundamental bases, and for that I am extremely grateful to everyone.

A special thanks goes to Stefania Serafin for welcoming me in Denmark (while expecting her little Emma), and for the bureaucratic-logistical-financial support during my period at the Aalborg University Copenhagen (July-December 2008).

The work contained in this thesis is the result of many contributions from different sources and extractions, and in this regard I wish to thank – in addition to those already mentioned – in a rigorously alphabetical order: Marco Civolani, Stefano Delle Monache (aka SteM), Antonio De Sena, Delphine Devallez, Carlo Drioli, Pietro Polotti and Stefano Zambon (aka SteZ).

Contents

Ringraziamenti V

Acknowledgments VII

Introduction 1

1 Study of a nonlinear impact model 9
1.1 Introduction 9
1.1.1 Impact modeling 9
1.1.2 Applications in acoustic modeling 10
1.1.3 Critical issues 11
1.2 The Hunt-Crossley impact model 11
1.2.1 Properties and analytical results 12
1.2.2 Addition of a constant external force 17
1.3 Numerical simulations 18
1.3.1 Numerical methods 18
1.3.2 Experimental results 20
1.4 Improved numerical simulations 25
1.4.1 Exploitation of analytical results 25
1.4.2 Numerical simulations with corrections 26
1.4.3 Computational cost 30
1.4.4 Evaluation of methods 32

2 Tools for ecologically-founded sound synthesis and design 35
2.1 Existing sound synthesis tools for interactive sonification 35
2.2 The Sound Design Toolkit 36
2.2.1 Introduction 37
2.2.2 The software package 37
2.2.3 A library of everyday sounds 39
2.2.4 Low-level models 40
2.2.5 High-level models 47

2.2.6 Advanced examples and sound palette . . . 59
2.2.7 User interface . . . 61

3 Applications in interactive sonification . . . 65
3.1 The Gamelunch . . . 66
3.1.1 Study of the dining scenario . . . 66
3.1.2 Enaction and sound feedback . . . 67
3.1.3 Realization . . . 68
3.1.4 Results . . . 71
3.2 DepThrow . . . 72
3.2.1 Architecture of the interface . . . 72
3.2.2 Components . . . 74
3.2.3 Objectives and issues . . . 75
3.3 Niw Shoes . . . 75
3.3.1 Interaction and physical design components . . . 76
3.3.2 Real-time feedback . . . 78
3.3.3 Goals and future work . . . 80

Conclusion . . . 83

A Numerical simulations of a modal resonator . . . 91
A.1 Modal resonators . . . 91
A.1.1 Continuous-time description . . . 91
A.1.2 Discretization . . . 92
A.2 Study of the harmonic oscillator . . . 94
A.2.1 Analytic time-domain solution . . . 94
A.2.2 Filter interpretation . . . 96
A.2.3 Discretization . . . 96
A.2.4 Remarks on the implementation efficiency . . . 98

References . . . 99

Introduction

The work presented in this thesis ranges over a wide variety of research topics, spacing from human-computer interaction to physical-modeling. What combines such broad areas of interest is the idea of using physically-based computer simulations of acoustic phenomena in order to provide human-computer interfaces with sound feedback which is consistent with the user interaction.

In this regard, recent years have seen the emergence of several new disciplines that go under the name of, to cite a few, *auditory display*, *sonification* and *sonic interaction design*. Hereafter, the fundamental notion and description of such disciplines are given, offering in this way the basics to better situate and understand the content of the thesis. Besides, the concepts of *physically-based sound synthesis* and *everyday sounds* are described.

This thesis deals with the design and implementation of efficient sound algorithms for interactive sonification. In particular, to this end, the physical modeling of everyday sounds is taken into account.

Interactive sonification

In most of the existing human-computer interfaces, communication and interaction still heavily rely on the visual channel, for example forcing us to constantly stare at displays. On the other hand, thanks to recent developments in information technology, new communication channels are being experimented with increasing interest, for example in the *virtual reality* (VR) community, or in the *ubiquitous computing* scenario, where computers are embedded in everyday objects. Within these contexts, auditory and tactile feedbacks have been especially recognized as playing an important role in information exchange between machines and individuals, and started to be integrated together with the visual channel in such a way that a genuine *multimodal* communication can be established. Such multimodal sensory integration can help to improve the user performance in accomplishing tasks. At the same time, multimodality is generally seen as a way of boosting the

user experience – in particular with respect to the *immersiveness*[1] – in such a way that the modalities of interaction with a device become closer to the usual ones we adopt in our everyday life to interact with the surrounding environment. Despite the fact that much research exists on how visual, auditory and tactile perception work when taken individually, still little is known about how humans actually integrate them. This is of course a matter of primary importance in order to develop truly multimodal interfaces, which are continuously and intuitively informative, as well as reactive and dynamic as the objects and environments which surround us.

Even when considered alone, auditory feedback offers many advantages and further possibilities as compared to its visual counterpart: indeed, sound does not require focused attention; it occupies the space without suffering from obstacle occlusion; it can convey multi-dimensional data streams that can be segregated by a human listener or, conversely, it enables a listener to evaluate many streams as a whole.

While research on interfaces relying on non-speech audio is still in its infancy, several important international initiatives already demonstrate an increasing interest in this area: for instance, the *International Community for Auditory Display* (ICAD)[2] is an international forum recognized by the National Science Foundation where subjects such as "sonification" or "auditory display" are regularly discussed. Also, it is worth mentioning that the discipline of *Sound and Music Computing* (SMC)[3] [88] has been recently included in the ACM Computing Classification System[4] as a sub-discipline of *Information Interfaces and Presentation*.

In order to better deal with this new research field, a few definitions are needed. According to Hermann[5] [56]:

> *Auditory displays* are systems where a human user makes sense of data using her/his listening skills, like for instance any data under analysis or data that represent states of the information processing system.
>
> *Sonification* is the use of sound – mainly non-speech audio signals – for representing or displaying data. Similar to scientific visualization, sonification aims at enabling human listeners to make use of their highly-developed perceptual (listening) skills for making sense of the data. More specifically, sonification refers to the technique used to create a sound signal in a systematic, well defined way, that involves data under examination as an essential ingredient for sound computing.
>
> *Interactive sonification* puts a particular focus on those systems where the user is tightly integrated into a closed-loop sonification system. [...] Interaction binds the user's actions to acoustic reactions, and thus creates modes of fast and seamless navigation and exploration, and ultimately, the experience of *flow* while performing an activity.

[1] A neologism commonly used within the VR and computer games communities to mean "the quality of engrossing or absorbing the user" (see for example `http://www.gamedev.net/reference/articles/article262.asp`).
[2] `http://www.icad.org`
[3] `http://www.smcnetwork.org/`
[4] `http://www.acm.org/about/class`
[5] The following quotations are taken from `http://www.sonification.de`

Despite the fact that several applications started taking advantage of auditory displays in such diverse fields as computer-assisted surgery, control of complex systems (for instance, aircraft cockpits, or control centers in nuclear reactors), analysis of massive datasets, however – as things stand now – most human-computer interfaces still do not fully exploit the great capabilities of human auditory perception, using instead the auditory channel only for basic and occasional static notification signals.

Sound synthesis

Generally, sound synthesis methods can be split in two broad classes [95]: the first one includes the so-called "signal models", whereas the second one includes "physically-based models" [23]. A brief description of these two approaches and a third, hybrid one, is reported hereafter:

Signal models have in common the fact that they represent sounds by describing their waveform. The latter is typically considered as a signal $s(t)$ in time or, equivalently, as a signal $S(s)$ in the frequency domain.
Signal methods do not deal with how a sound has been generated from its source and possibly processed by other factors, instead they only consider the result that the listener can hear. Consequently, since they generally act on quantities (like frequency, amplitude, etc.) which do not have a natural[6] interpretation, the corresponding synthesis algorithms and their control parameters feel like abstract, and do not allow the user to get an intuitive handling of the sound phenomena.
The corresponding algorithms are usually simple and computationally efficient, and suit the generation of sounds of very diverse nature. This partly explains why – up until recently – signal models have been the ones to be preferred. Nevertheless, the current standard for sonification in human-computer interfaces – that is, the playback of sound samples – is increasingly being recognized as not satisfactory for its static, repetitive, not reactive nor dynamic character.
Examples of methods in this class are: FM, additive, wavetable and sample-based syntheses.

Physically-based models give an account of objects and interactions which result in the production of sound, instead of describing the sound itself. In other words, the generation mechanics behind a sound (that is, the *cause*) is physically described in the shape of models which are then translated into algorithms. Consequently, sound (that is, the *effect*) is obtained as a result of running numerical simulations of such models.
Compared to signal-based models, these use much less memory (or bandwidth), even though they usually require much more computational power. Also, since they model specific acoustic phenomena, they are in a sense less versatile: in other words, generally a physically-based model cannot render a wide range of

[6] Or "ecological". More later on this.

sound classes (for example impacts, frictions, liquid-related sounds, etc.), but only – though very expressively – the class of sounds it was designed for.
Because of the physical consistency of the models, their control parameters have direct correspondence with physical quantities (for instance, velocity, force, displacement, etc.), and result in physically-consistent sonic behaviors (for example, increasing the velocity of impact results in both a loudest and brightest sound).
Moreover, such models can be used – though only to some extent – for driving graphic[7] and haptic rendering together with sound synthesis: that would allow to obtain a strict coherence between visual, tactile and sound events (multimodality) [5, 33, 54, 76, 111].

Physically informed/inspired models represent a hybrid approach that can make use of disparate synthesis methods (even signal-based) and do not aim at modeling excitation and vibration phenomena as thoroughly as in pure physically-based models. Still, they exploit physically-consistent models of varied complexity and on different levels.
As an example, models of non-acoustic physical phenomena can be used to drive the control layer of a sound synthesis engine, thus enabling such hybrid model to expose control parameters which correspond to physical quantities and reflect physically-consistent sonic behaviors.

Everyday-sound models for interactive sonification

From the previous section it is clear that physically-based (or -informed/-inspired) sound models are a very suitable choice for addressing real-time interactive applications. Indeed, in such applications the potential interaction is far too open-ended for all possibilities to be, for example, pre-rendered as audio samples.

In this regard, in [57] a paradigm called *Model Based Sonification* (MBS) is proposed: according to this approach, sound feedback is the physically coherent result of physically modeled excitation-resonance processes. As an example, the user excites a sonification model, and such interactions cause auditory responses that depend upon the underlying data, thus representing information about them.

In particular, the increasing availability of low-cost embeddable CPUs, actuators, and sensors which can be used to link the models' control parameters to gestures, makes physically-based sound synthesis a promising paradigm for the design of interactive continuous sound feedback in the everyday context. However, it must be said that the design of effective mappings between the user's gesture and sound control parameters constitutes a crucial issue.

Another central issue in sonification concerns the choice of suitable, expressive and meaningful sounds. Starting from the early eighties, the ecological school of psychoacoustics [48, 114] pointed out the importance of human auditory perception as conveying information about processes in our everyday surroundings. Moreover,

[7] Since the physics of light is very different from that of mechanical interactions [69], in this case many would be the limitations of such extension.

it was observed [47] that in *everyday listening* we perceive sound *sources* rather than abstract, signal-based attributes as in *musical listening*.

In this perspective, *everyday sounds* can be regarded as the natural sonification of information about our environment. For this reason, the use of expressive and meaningful everyday sounds as sound feedback in human-computer interfaces has several advantages, most notably: it allows to exploit a "language" which is universally and naturally interpreted by humans, and therefore virtually no explanation and learning are necessary for the user; the sonified device is acoustically merged with the environment, in this respect "ecologically disappearing".

A deeper, dedicated link which joins the experience of physically-based sound synthesis – also taking into account aspects of usability and implementation – and the insights of ecological psychoacoustics, has only recently started to develop.

Past and current research projects

In this section, some current and past research projects are briefly reviewed, which in different ways address topics related to the synthesis of everyday sounds and interactive sonification in everyday contexts.

The EU-funded project "the Sounding Object" (*SOb*, 2001-2003) [97] was launched as part of the "Disappearing Computer" initiative of FET-Proactive, to provide a corpus of knowledge in everyday-sound perception, accompanied by suitable new methods and tools for physically-based sound modeling and for high-level control of such models. Partners in the project included experts in digital signal processing, musical acoustics, experimental psychology and human-computer interaction. The project *SOb* aimed at *sounding objects* [96] that incorporate a (possibly) complex responsive acoustic behavior, and that are expressive in the sense of ecological hearing. Realistic sound simulation was not necessarily the perfect goal: simplifications which preserved and possibly exaggerated certain acoustic aspects, while losing others considered less important, were often preferred. Besides being more effective in conveying certain information, such "cartoonifications" are often cheaper to implement, just like graphical icons are both clearer and easier to draw than photo-realistic pictures. Also, control models were developed, for example driving the models' parameters by means of preset trajectories, or mapping them to gestures. Finally, the models were tested by means of perceptual validation, for instance doing listening tests, or asking participants to characterize a class of sound events.

The EU-funded project "Closing the Loop of Sound Evaluation and Design" (*CLOSED*, 2006-2009)[8] [105] was started as part of the NEST-Path "Measuring the Impossible" activity, to address some critical issues related to sound evaluation and design. In particular, the project sprang from the observation that sound design is not yet a solid discipline, and the main reasons for that were found in the lack of design-oriented measurement and evaluation tools. Toward closing the loop of sound evaluation and design, the project aimed at providing functional/aesthetic sound design tools suitable for interactive scenarios and effectively usable by designers, linking physical attributes of sound-enhanced everyday

[8] `http://closed.ircam.fr/`

objects to emotional responses. Other goals were: to provide an updated corpus of knowledge on everyday-sound perception and categorization, to develop effective methodologies for interactive sound design, and to provide machine learning tools for perceptually-based evaluation of the sound design process. As a example of practical utilization of the project's outcomes, it can be noted that while every child knows how to draw "smileys" or other "cartoon" icons, basic methodologies to approach similarly ecologically expressive sound design are still lacking.

The discipline of *sonic interaction design* [98, 99] refers to the exploitation of sound as one of the principal channels conveying information, meaning, and aesthetic/emotional qualities in interactive contexts. The current COST Action "Sonic Interaction Design" (*SID*)[9] was started in 2007 with the aim of strengthen the links between scientists, artists and designers, in order to contribute to the creation and consolidation of new design theories, tools, and practices in this innovative and interdisciplinary domain. One of the most important objectives in *SID* is to understand how interaction in sonification can support the utility, effectiveness, acceptance and perceived aesthetics of auditory displays.

The EU-funded project "Natural Interactive Walking" (*NIW*)[10] was started in 2008 as part of the FET-Open activity, to investigate the possibilities of using haptic and auditory modalities in floor interfaces, and the synergy of perception and action in capturing and guiding human walking. To this end, *NIW* also draws inspiration from current studies in sonic interaction design and interactive sonification, while providing opportunities to set new results in these fields of research. As one of outcomes of the project, floor- and shoe-based interfaces are being designed and prototyped, and their applicability to, for instance, navigation aids, augmented reality and rehabilitation, are currently being investigated.

Outline

The thesis can be seen as divided in two parts of different nature: one dealing with theoretical aspects (Chapter 1 and Appendix A), and one illustrating a process of development, implementation and application in the context of human-computer interaction (Chapters 2 and 3).

In more detail, in Chapter 1 a non-linear impact model is reviewed using analytical tools (Section 1.2) and then discretized by means of several numerical methods (Section 1.3.1).

The resulting numerical systems are studied and compared in the light of their accuracy and efficiency (Section 1.3.2): it is shown that, for some regions of the parameter space, the trajectories of the discretized systems may significantly drift from the analytically derived curves, resulting in an inconsistent energy content (Sections 1.4.1).

Energy inconsistencies can give rise to instabilities and compromise the realistic behavior of the system, especially when spurious energy is generated, thus losing the property of passivity. In order to handle these issues, some corrections

[9] `http://www.cost-sid.org/`
[10] `http://www.niwproject.eu/`

are suggested that allow to improve the accuracy and stability of the numerical simulations, moreover restoring their energy behavior (Section 1.4.2).

Finally, the numerical systems obtained with and without corrections are compared in terms of their computational complexity. It is shown that it is possible to implement very accurate, stable yet efficient numerical simulations (Sections 1.4.3 and 1.4.4).

In Chapter 2, an open-source software product for ecologically-founded sound synthesis and design is described: the Sound Design Toolkit (SDT) offers state-of-the-art physically-consistent tools for designing, synthesizing and manipulating ecological sounds in real-time. The aim of the SDT is to provide efficient and effective instruments for interactive sonification and sonic interaction design.

The development, content, structure and distribution of the package are described in Section 2.2.2.

The sound and control algorithms provided in the SDT are categorized in low-level and high-level models, respectively depicted in Section 2.2.4 and 2.2.5. In particular, several models of contact between solids (e.g., impact, friction, rolling, crumpling, etc.) and simulations of acoustic phenomena related to liquids and gases are provided (e.g., dripping, boiling, splashing, etc.). Also, high-level models include algorithms for the control and layering of the low-level ones.

Section 2.2.6 describes some advanced examples and tools supplied with the SDT, which enable to exploit the models to their full potential. Among other things, a collection of presets (that is, recallable sets of parameters for the models) are provided, which show the wide and expressive sound palette achievable with the SDT.

Finally, in Section 2.2.7 the SDT's help system together with a recently introduced centralized front-end interface are described. The latter offers many facilities in terms of connectivity, mapping and usability of the models, thus making the SDT an accessible and flexible tool for sound designers, and facilitating its use in real-time interactive applications.

Chapter 3 illustrates three major prototype applications in interactive sonification which have been developed.

The Gamelunch (Section 3.1) is a sonically-augmented dining table, which was originally developed within the project *CLOSED*. The main aim of the Gamelunch is to let users experience the importance of interaction-coherent sounds in everyday-life acts, as that of sitting at a table and having lunch. This application exploits the power and flexibility of physically-based sound models towards the investigation of the closed loop between interaction, sound and emotion.

DepThrow (Section 3.2) is an interactive audio-only game also developed within the project *CLOSED*, and designed around the auditory perception of distance and the use of physically-based models for the simulation of the dynamics, sound source, and acoustical environment. The user performance is closely related to her/his ability to perceive the dynamic distance of a virtual sounding object. Therefore this game represents a potential tool for exploring the usability of auditory distance information in interaction design.

The Niw Shoes (Section 3.3) are a wearable shoe-based prototype interface developed for the project *NIW*, which provides the user with audio-haptic cues of ground. Thanks to the interface, neutral floors can be interactively augmented so as to react like they were made of a different material. The potential advantages of wearing actuated shoes are manifold: some example scenarios where they could be profitably used include augmented and virtual realities, navigation aids and rehabilitation.

Lastly, in Appendix A some preliminary results of an ongoing study on the numerical simulation of the harmonic oscillator are described. The aim of the study is to provide a better realization of the modal resonator, as compared to that already included in the SDT, especially regarding its computational efficiency and energy accuracy.

1

Study of a nonlinear impact model

In this chapter, a physically-based impact model – already known and exploited in the field of sound synthesis – is studied using both analytical tools and numerical simulations. It is shown that, for some regions of the parameter space, the trajectories of the discretized systems may significantly drift from the analytically-derived curves.

The energetically-consistent and phenomenologically-plausible behavior of contacting bodies is especially crucial in simulations of interactions based on sustained or repeated impacts, such as in rolling, crumpling, or bouncing.

In this regard, some methods are proposed, based on enforcing numerical energy consistency, which allow to improve the accuracy and stability of numerical simulations.

1.1 Introduction

1.1.1 Impact modeling

Physical models of impacts between objects are ubiquitous in many areas of science and engineering, including robotics [71], haptics [70], computer graphics [74], acoustics [16] and sound synthesis [7].

Impact may be defined as a sudden change in the momentum of each contacting body, without a corresponding change in position [38]. The common goal of impact models is to be able to predict the behavior of colliding objects. In what follows, only impacts between rigid bodies are considered.

In particular, different approaches or aspects can be found in impact dynamics literature [38]:

- the *Classical Mechanics* approach [53], whose core is the impulse-momentum law, and where the loss of energy occurring during an impact is taken into account by means of the *coefficient of restitution*, thus allowing to predict the velocities after contact. This approach is unable to predict the contact force between bodies or the stresses in them;
- when vibrational dissipation is an important fraction of the total energy, the classical approach becomes insufficient to examine an impact problem, and the *elastic stress wave propagation* theory [53] should be taken into account.

- the *Contact Mechanics* approach [53], which is conventionally mainly concerned with static contact, although it has been extended to approximate solutions when impact is involved. For example, the Hertz model of collision between two spheres is used to obtain the force deformation relation needed to calculate the duration of impact and the maximum indentation. Such models can be extended to include internal viscosity, thus allowing to effectively model the contact area as a spring-damper system;
- when plastic strains go beyond the scale of contained deformation, the elastic wave propagation model can no longer be applied to analyze impact problems, and the *plastic deformation* theory should be considered [53]. This is the domain of high velocity impact generally associated with explosives and projectiles.

In interactive contexts it is often necessary to treat impacts as continuous-time dynamic phenomena, and impact models should be used that are able to describe the evolution of forces at the contact point. In order to obtain impact durations greater than zero, interacting bodies should be modeled as compliant objects, that is they deform during impact. For the reasons pointed out above, and considering that the wave propagation theory can result in extremely complex systems, in what follows only the Classical Mechanics approach is taken into account. Also, since high velocity impacts are out of the scope of this chapter, plastic deformations will not be considered.

In the extended Hertz model [40], where internal viscosity is taken into account, the restitution force is the sum of a nonlinear elastic term – in the form of a power law of compression – and a dissipative component proportional – via a second power law of compression – to the compression velocity. The exponents of the two power laws, as derived for two colliding balls, are 3/2 and 1/2, respectively [65].

An impact model introduced by Hunt and Crossley [32,39,61,66,71] generalizes the extended Hertz model by considering a variable exponent that accounts for different shapes or, equivalently, for the growth rate of the contacting surfaces. In this model the power laws in the elastic and dissipative term are considered to be equal, thus allowing easier closed-form calculations [89]. The Hunt-Crossley model is consistent with the notion of *coefficient of restitution* used to characterize energy loss during impacts [32].

In the context of musical acoustics, Stulov proposed a piano hammer model including the relaxation properties of felt [104]. Such model has exponents α and $\alpha - 1$ for the power laws, and the actual value of α can be used to match experimental data.

Other models exist that take plastic deformations into account, thus introducing abrupt direction changes in the force-compression curves at the transition between loading and unloading [118].

1.1.2 Applications in acoustic modeling

Contact models can serve as a basis for developing models of acoustic phenomena. In the context of physically-based sound synthesis, the Hunt and Crossley model has been used to develop an impact sound model [7], where a generic resonating object is used in place of the classic rigid wall.

Other models of more complex acoustic phenomena have been developed based on the very same impact model studied here. For instance, a *rolling* sound model has been implemented by driving an impact sound model by means of a physically-inspired control layer. More precisely, the continuous interaction of a ball rolling on a surface has been modeled as a dense temporal sequence of micro-impacts driven by the geometry of the contacting surfaces, and modulated by the ball's asymmetry. Further details and more examples of sound models based on the Hunt and Crossley impact model are provided in Section 2.2.5.

In the context of musical sound synthesis, the piano and other percussive musical instruments have also been modeled by using dissipative impact models [104].

1.1.3 Critical issues

Any numerical system can simulate its continuous-time counterpart only up to a limited accuracy. One of the basic reasons for this inherent limit is that the discretized variables depend on the chosen *sample rate* F_s [1, 55], and therefore the accuracy of any numerical method depends on it.

Apart from this intrinsic limitation, it must be noted that different numerical methods behave differently, and that all of them are susceptible to problems related to stability. For instance, for some system parameter values, one method could dissipate too much energy, while another one could violate the principle of passivity, thus generating spurious energy.

Even though some simple stability conditions can be derived for LTI systems (for example, the von Neumann analysis [90]), these do not extend to the case of nonlinear systems. In this latter case, one possibility is to exploit methods based on principles of conservation/dissipation of energy (see for example [12,109] in the context of physical modeling sound synthesis).

As for applications which make use of impact/contact models, energy inconsistencies are a recurring issue. In computer graphics, where the constraint of low frame rates makes numerical systems prone to instabilities [74], a typical example is provided by a steady object in sustained contact with a rigid floor: when the system does not retain passivity, the object can move upward and bounce. Similar issues are encountered in simulations of haptic contact, where stiffness values are usually limited by requirements on system passivity [17,31,37,64], whereas higher values can cause the system to become unstable, for example entering an oscillatory regime, or reacting actively to the input. In numerical sound synthesis [13] by physical models, artifacts and inconsistencies can become audible especially in situations of sustained or repeated (see Section 1.4.2) contact interactions, as in rolling, sliding or bouncing.

1.2 The Hunt-Crossley impact model

The Hunt-Crossley impact model [61] is defined by the following nonlinear equation describing the impact force:

$$f(x,v) = \begin{cases} kx^{\alpha} + \lambda x^{\alpha} v = kx^{\alpha} \cdot (1 + \mu v) & x > 0 \\ 0 \quad , & x \leq 0 \end{cases} \tag{1.1}$$

where x is the *compression*, $v = \dot{x}$ is the *compression velocity*, $\alpha > 1$ is the exponent of a power law and represents the *local shape* of contact surfaces, k is the *stiffness coefficient*, and $0 \leq \lambda \leq k$ is the *damping coefficient*. The mathematically convenient term μ $(= \lambda/k)$ allows to simplify some closed-form calculations. The impact force model thus represents a nonlinear spring of constant k in parallel with a nonlinear damper of constant λ. The term kx^α corresponds to the *elastic component*, while $\lambda x^\alpha v$ represents the *dissipation* due to internal friction.

Marhefka and Orin [71] made use of the Hunt-Crossley model in order to represent the impact between a lumped point-mass and a rigid wall (representing a comparatively massive surface which does not move during collision), therefore considering the system described by the equation:

$$ma(t) = -f(x(t), v(t)) \tag{1.2}$$

where m is the *mass*, and a is its *acceleration*. In this basic case, during contact the compression and the compression velocity are respectively equivalent to the displacement and the velocity of the point-mass.

1.2.1 Properties and analytical results

Thanks to the simple form of (1.2), the model can be treated analytically and some of its properties can be inferred. Hereafter the initial conditions $x(0) = 0$ and $\dot{x}(0) = v_{\text{in}}$ are considered, that is to say that the point-mass hits the rigid wall with velocity v_{in} at time $t = 0$.

Compression

It is shown in [71] that from (1.2) it follows:

$$x(v) = \left[\frac{m(\alpha+1)}{k\mu^2} \cdot \left(-\mu(v - v_{\text{in}}) + \log\left|\frac{1+\mu v}{1+\mu v_{\text{in}}}\right|\right)\right]^{\frac{1}{\alpha+1}} \tag{1.3}$$

which can be exploited for plotting the phase portraits on the (x, v) plane shown in Fig. 1.1. From Fig. 1.1 it can be inferred that, due to the viscous dissipation occurring during contact, the relation $v(t + dt) < v(t)$ holds, and in particular the output velocity v_{out} is always smaller in magnitude than the corresponding v_{in}. Moreover, for increasing v_{in}'s, v_{out} converges to the limit value $v_{\text{lim}} \triangleq -1/\mu$. The line $v = v_{\text{lim}}$ represents the trajectory where the elastic and dissipative terms cancel, and separates two regions of the phase space, each of which is never entered by trajectories started in the other one.

Equation (1.3) allows to infer the *maximum compression* experienced during contact, which occurs when the compression velocity equals zero:

$$x_{\max} = x(0) = \left[\frac{m(\alpha+1)}{k\mu^2} \cdot \left(\mu v_{\text{in}} + \log\left|\frac{1}{1+\mu v_{\text{in}}}\right|\right)\right]^{\frac{1}{\alpha+1}}. \tag{1.4}$$

As remarked in [71], (1.1) together with Fig. 1.1 show that the force f becomes sticky (inward) when $v < v_{\text{lim}}$. However there is no physical inconsistency in

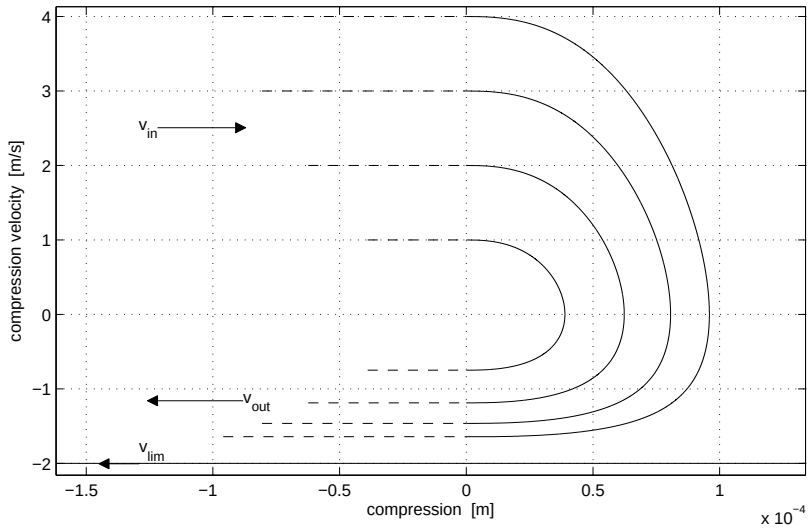


Fig. 1.1. Phase portraits for varying input velocities: $v_{\text{in}} = 1 \dots 4$ m/s. Other values of parameters are: $m = 10^{-2}$ kg, $k = 10^9$ N/m$^\alpha$, $\mu = 0.5$ s/m, $\alpha = 1.5$. Solid lines represent the trajectory of the mass during contact; dashed lines represent free motion.

this "stickiness" property, and indeed this never occurs for trajectories with initial conditions $x(0) = 0$ and $\dot{x}(0) = v_{\text{in}} > 0$.

Finally, by substituting (1.3) in (1.2) one can plot the compression-force characteristics during collision, which are shown in Fig. 1.2. It can be noted that the dissipative term $\lambda x^\alpha v$ introduces hysteresis around the curve kx^α. Also, it is worth noticing that there are no discontinuities in the impact force curve, even at transitions from contact to non-contact states and vice versa.

Output velocity

The *restitution coefficient* E is defined as:

$$E \triangleq \left| \frac{v_{\text{out}}}{v_{\text{in}}} \right| . \tag{1.5}$$

Note that v_{in} and v_{out} correspond to the roots of the right-hand side of (1.3), that is the points where $x = 0$. As a result, v_{out} can be defined implicitly from (1.3) as a function of (μ, v_{in}) only:

$$\mu v_{\text{out}} - \log|1 + \mu v_{\text{out}}| = \mu v_{\text{in}} - \log|1 + \mu v_{\text{in}}| \ . \tag{1.6}$$

This implies that μv_{out} is a function of μv_{in} only, and therefore E is also a function of μv_{in} only.

Analytical derivations of the dependence $E(\mu v_{\text{in}})$ have been classically performed in the limit of small initial velocities and/or small dissipation [61]. For instance, Hunt and Crossley found that, in the limit $\mu v_{\text{in}} \to 0_+$, the restitution

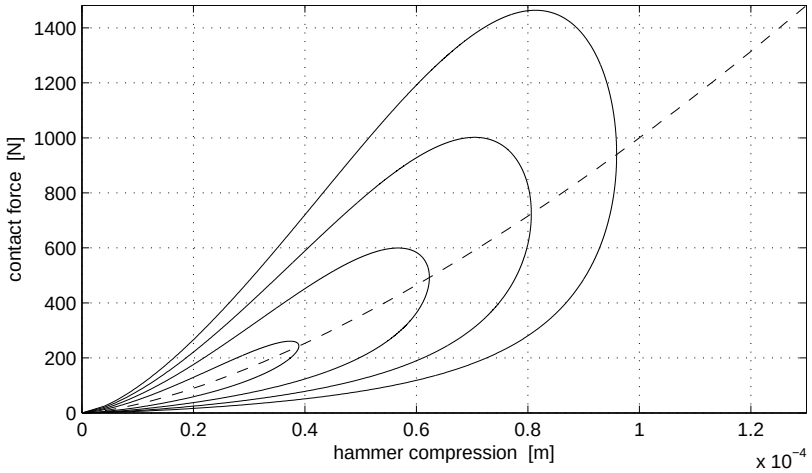


Fig. 1.2. Compression-force characteristics for varying input velocities: $v_{\text{in}} = 1 \dots 4$ m/s. Solid lines represent the case when dissipation is taken into account (the values of parameters are the same as in Fig. 1.1), while the dashed line represents the curve kx^{α}, where no dissipation is considered ($\lambda = 0$).

coefficient can be approximated as the linear function $E(\mu v_{\text{in}}) = (1 - \frac{2}{3}\mu v_{\text{in}})$, from which it follows:

$$v_{\text{out}} = -v_{\text{in}} \cdot (1 - \frac{2}{3}\mu v_{\text{in}}), \quad \mu v_{\text{in}} \to 0_+ \tag{1.7}$$

where it was considered that $\text{sgn}(v_{\text{out}}) = -\text{sgn}(v_{\text{in}})$. However in [80, 81] it is suggested that a non-local approximation $\tilde{v}_{\text{out}}$ can be empirically determined as an *ansatz* which fits the curve $E(\mu v_{\text{in}})$ in the two limit regions $\mu v_{\text{in}} \to 0_+$ and $\mu v_{\text{in}} \to +\infty$, thus obtaining:

$$\tilde{v}_{\text{out}}(\mu, v_{\text{in}}) = v_{\text{lim}} \left[1 - \left(\sum_{j=0}^{n} b_j \cdot v_{\text{in}}^j \right) e^{-2\mu v_{\text{in}}} \right] \tag{1.8}$$

where, in the case $n = 4$, the coefficients b_j are:

$$b_0 = 1, \quad b_1 = \mu, \quad b_2 = \frac{2}{3}\mu^2, \quad b_3 = \frac{2}{9}\mu^3, \quad b_4 = \frac{14}{135}\mu^4. \tag{1.9}$$

From now on, unless specified otherwise, the notation $\tilde{v}_{\text{out}}$ will refer to the fourth-order approximation provided by (1.8) and the coefficients (1.9).

It is worth noticing that conventional iterative zero-finding methods applied to (1.6) can always be used to compute a more precise release velocity at a higher computational cost (see Section 1.4.3). Fig. 1.3 and Table 1.1 show the error introduced by the approximate value $\tilde{v}_{\text{out}}$, when compared to the corresponding value computed numerically as a zero of (1.6).

Contact time

It is shown in [9] that the *contact time* can be expressed as:

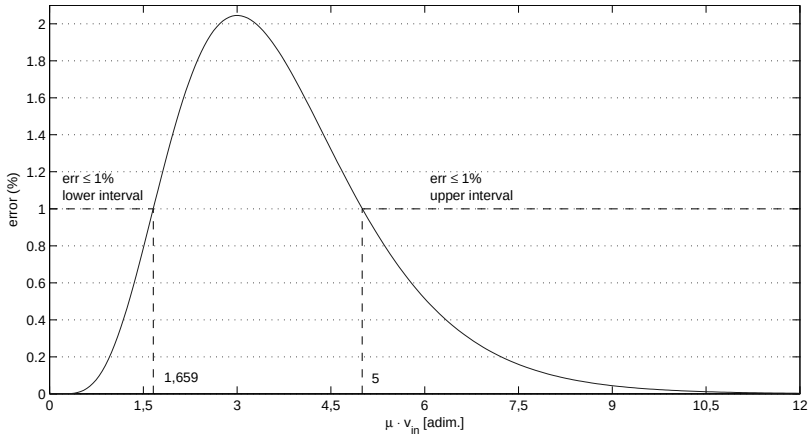


Fig. 1.3. Percentage error on the output velocity introduced by $\tilde{v}_{\text{out}}$ with respect to the value computed numerically as a zero of (1.6). As an example, the ranges of μv_{in} are shown, for which the maximum error is equal to 1% (see Table 1.1).

Table 1.1. Summary of errors introduced by $\tilde{v}_{\text{out}}$ for different ranges of μv_{in}.

%err $\tilde{v}_{\text{out}}$	$\leq 1\%$	$\leq 0.1\%$	$\leq 0.01\%$	$\leq 0.001\%$
μv_{in}	≤ 1.659	≤ 0.798	≤ 0.476	≤ 0.319
	≥ 5	≥ 8.054	≥ 10.660	≥ 13.170

$$\tau = \left(\frac{m}{k}\right)^{\frac{1}{\alpha+1}} \cdot \left(\frac{\mu^2}{\alpha+1}\right)^{\frac{\alpha}{\alpha+1}} \cdot \int_{v_{\text{out}}}^{v_{\text{in}}} \frac{dv}{(1+\mu v)\left[-\mu(v-v_{\text{in}}) + \log\left|\frac{1+\mu v}{1+\mu v_{\text{in}}}\right|\right]^{\frac{\alpha}{\alpha+1}}} \,. \tag{1.10}$$

Equation (1.10) states that the contact time τ depends only on μ, the exponent α and the ratio m/k, plus obviously the impact velocity v_{in}. Since neither m nor k affect the value of the integral (recall that v_{out} depends only on μ and v_{in}), it follows that, given a fixed v_{in}, the proportionality $\tau \sim (m/k)^{1/(\alpha+1)}$ holds.

From an auditory point of view the value of the contact time is strongly correlated to the perceived "hardness" of the impact [9, 51]. Namely, as the contact time decreases, the perceived hardness increases. Recalling the power-law dependence above and (1.1) it follows that, for a fixed mass m, "hard" and "soft" impacts correspond respectively to high and low force values.

Energy properties and behavior

The energy variation in a mechanical system can be calculated as the work made by the non-conservative forces f_{nc} acting on the system along a certain path $x_1 \to x_2$:

$$\Delta H = \int_{x_1}^{x_2} f_{\text{nc}}(x)\, dx = \int_{t_1}^{t_2} f_{\text{nc}}(t) v(t)\, dt = -\Delta\Lambda \tag{1.11}$$

where H is the total energy content known as the *Hamiltonian*, Λ is the energy dissipation, and the second integral is obtained by considering that t_1 and t_2

correspond respectively to the instants when the displacements x_1 and x_2 are reached. The Hamiltonian H is the sum of potential and kinetic energies, hereafter named V and T, respectively:

$$H(t) = V(t) + T(t) \ . \tag{1.12}$$

With regard to the system represented by (1.2), T is related to the dynamics of the point-mass, which is described by the left-hand side of (1.2), while V is related to the elastic component of the impact force defined in (1.1).

In agreement with the last integral in (1.11), multiplying both sides of (1.2) by $v(t) = dx/dt$ and integrating them over time, gives:

$$\underbrace{\int_0^t ma(s)v(s)\ ds}_{T(t)-T_0} = -\underbrace{\int_0^t kx(s)^\alpha v(s)\ ds}_{V(t)} - \underbrace{\int_0^t \lambda x(s)^\alpha v(s)^2\ ds}_{\Lambda(t)>0} \tag{1.13}$$

where the force expression of (1.1) has been considered in the case $x > 0$ only. The first two integrals in (1.13) can be solved explicitly, obtaining:

$$V(t) = \frac{kx(t)^{\alpha+1}}{\alpha+1} \ , \quad T(t) = \frac{mv(t)^2}{2} \ . \tag{1.14}$$

Considering a system where the point-mass travels with velocity v_{in} before an impact occurs, then the *initial Hamiltonian* corresponds to the initial kinetic energy:

$$H_0 = T_0 = \frac{mv_{\text{in}}^2}{2} \ . \tag{1.15}$$

From (1.11) it follows that at each time instant t the current energy content is:

$$H(t) = H_0 - \Lambda(t) \tag{1.16}$$

and, since $\Lambda(t) > 0$, the following inequalities hold:

$$0 \le H(t + dt) \le H(t) \ . \tag{1.17}$$

Indicating τ as the instant when an impact ends, the *final Hamiltonian* of the system – that is the energy content right after contact – can be written as:

$$H_\tau = T_\tau = \frac{mv_{\text{out}}^2}{2} \ . \tag{1.18}$$

Also, the total amount of energy dissipation occurred during contact is:

$$\Delta H_\tau = H_\tau - H_0 = -\Lambda_\tau \tag{1.19}$$

which corresponds to the area enclosed by the hysteresis loops shown in Fig. 1.2.

As for the rightmost integral in (1.13) – which is unsolvable – an equivalent expression can be obtained by substituting the complementary results for the remaining integrals:

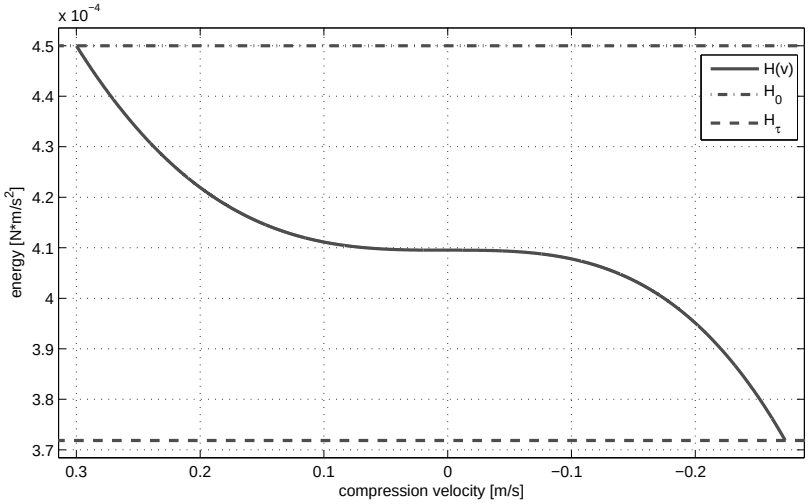


Fig. 1.4. Compression velocity-Hamiltonian characteristic. The two horizontal lines display $H_0 = T_0$ and $H_\tau = T_\tau$, that is respectively the initial and the final Hamiltonian. The values of parameters are the same as in Fig. 1.1 for $v_{\text{in}} = 1$ m/s. Notice that the compression velocity axis has been inverted, thus allowing to read the graph from left to right.

$$\Lambda(t) = \int_0^t \lambda x(s)^\alpha v(s)^2 \, ds = \frac{m(v_{\text{in}}^2 - v(t)^2)}{2} - \frac{kx(t)^{\alpha+1}}{\alpha+1} \, . \tag{1.20}$$

Finally, by substituting (1.3) in (1.20) and recalling (1.16), the following expression in v only is found [80]:

$$H(v) = H_0 - \Lambda(v) = \frac{m}{2}v^2 - \frac{m}{\mu}(v - v_{\text{in}}) + \frac{m}{\mu^2}\log\left|\frac{1+\mu v}{1+\mu v_{\text{in}}}\right| \tag{1.21}$$

which can be used for plotting the curve shown in Fig. 1.4.

The results of (1.20) and (1.21) together set a considerable improvement with respect to the corresponding expressions found in [32]: in that work only approximate results are provided, which have been obtained initially considering the simplified case where $\lambda = 0$, and then extending them to the general case.

1.2.2 Addition of a constant external force

When a constant external force f_e (for instance, the force of gravity) is applied to the point mass, (1.2) has to be rewritten as:

$$ma(t) + f_e = -f(x(t), v(t)) \, . \tag{1.22}$$

Unfortunately, in this case no closed-form analytical results can be found as those described in Section 1.2.1. More specifically, multiplying both sides of (1.22) by $v(t)$ and integrating them over time, an equation is found where an unsolvable integral is present, this way preventing to obtain explicit-form expressions for $x(v)$, $\tilde{v}_{\text{out}}$ and τ.

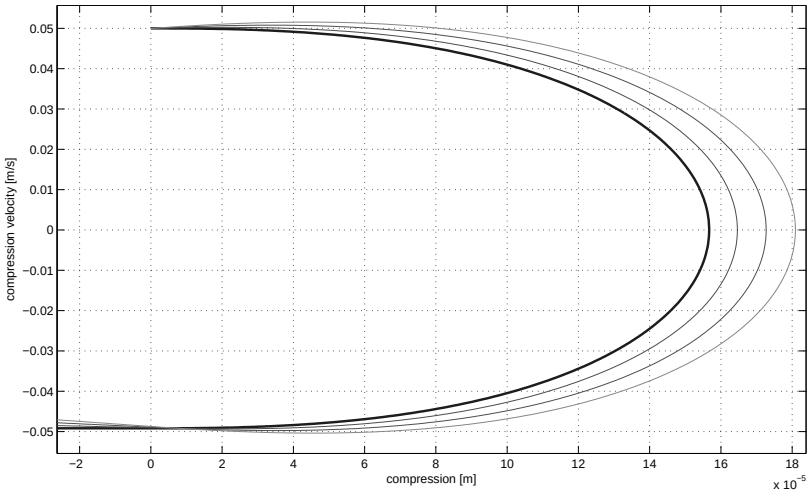


Fig. 1.5. Phase portraits of impacts for different external forces applied: $f_e = 1 \dots 3 \cdot m$ N. The bold line represents the case where $f_e = 0$. The values of parameters are the same as in Fig. 1.1 except for $k = 10^5$ N/m$^\alpha$ and $v_{\text{in}} = 0.05$ m/s. Notice that, due to the lack of analytical results when an external force is present, the corresponding phase portraits have been obtained as the result of highly oversampled numerical simulations.

Rewriting the 2nd-order equation (1.22) as a system of 1st-order equations:

$$\begin{cases} \dot{x} = v \\ \dot{v} = -\frac{k}{m} x^\alpha - \frac{\lambda}{m} x^\alpha v - \frac{f_e}{m} \end{cases} \tag{1.23}$$

the equilibrium point of the system is found to be $(x_{\text{eq}}, v_{\text{eq}}) = (-f_e^{1/\alpha}/k, 0)$, which corresponds to the *compression offset* in stationary conditions.

As Fig. 1.5 shows, for positive values of f_e and v_{in}, the velocity of the point mass during the compression phase is generally greater than in the case when $f_e = 0$. In particular, at the beginning of contact interaction, since f_e is generally higher than the current impact force f, the compression velocity exceeds v_{in}. On the other hand, compared to the case when $f_e = 0$, during the decompression phase the absolute value of the point mass velocity v decreases, resulting in lower output velocities. Moreover, the resulting maximum compression is always greater than that calculated by (1.4).

1.3 Numerical simulations

In this section, the continuous-time system described by (1.2) is discretized by means of several numerical methods, and the resulting numerical systems are studied.

1.3.1 Numerical methods

Different numerical methods were considered, which are commonly used in various fields of applications spacing from computer graphics, to physical simulation of

dynamical systems and digital signal processing. The methods were chosen for their low order, which generally results in low computational cost, thus being suitable for real-time applications.

Following a standard notation in numerical analysis, hereafter the integration step is a constant named h $(= 1/F_s)$.

1-step Adams-Moulton (AM1)

The *1-step Adams-Moulton* (AM1) [90] is a A-stable 2nd-order implicit method, also known as *bilinear transformation*, or *trapezoidal rule*.

Discretizing (1.2) results in the following equation in state-space form:

$$\begin{bmatrix} x_{n+1} \\ v_{n+1} \end{bmatrix} = \begin{bmatrix} 1 & h \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_n \\ v_n \end{bmatrix} + \begin{bmatrix} \frac{h^2}{4m} \\ \frac{h}{2m} \end{bmatrix} [f_{n+1} + f_n] \tag{1.24}$$

where the expression for the discrete-time force is obtained by replacing the continuous-time variables $x(t)$ and $v(t)$ in (1.1) with their discrete-time counterparts.

Since AM1 is an implicit method, (1.24) is also in implicit form, and this is reflected in the instantaneous relationship between $[x_{n+1}\ v_{n+1}]^T$ and f_{n+1}. Unfortunately, since f_{n+1} also has an instantaneous dependence on x_{n+1} and v_{n+1} (given by (1.1)), the discrete-time counterpart of the system described by (1.2) contains a *delay-free loop*, which is not directly computable and – because of the nonlinear dependence $f(x, v)$ – needs some special handling in order to be solved. In particular, the *K-method* [14] together with *Newton's method* [90] are used, weighing on the efficiency of the simulation (see Section 1.4.3).

Velocity Verlet

The *Verlet method* [115] is a 2nd-order explicit method commonly used in computer graphics [74], video games, and molecular dynamics simulation, where it is typically used for integrating Newton's equation of motion in order to describe the trajectory of moving particles. The one used here is a variant, called *velocity Verlet*, which provides better handling of the velocity variable and can be seen as a predictor-corrector method.

Discretizing (1.2), results in the following implementation scheme:

$$\begin{aligned} x_{n+1} &= x_n + hv_n + \frac{h^2}{2}\frac{f_n}{m} \ , \\ v_{n+\frac{1}{2}} &= v_n + \frac{h}{2}\frac{f_n}{m} \ , \quad \text{predictor,} \\ f_{n+1} &= f(x_{n+1}, v_{n+\frac{1}{2}}) \ , \\ v_{n+1} &= v_{n+\frac{1}{2}} + \frac{h}{2}\frac{f_{n+1}}{m} \ , \quad \text{corrector.} \end{aligned} \tag{1.25}$$

It should be noted that this algorithm assumes that f_{n+1} only depends on the predicted velocity $v_{n+\frac{1}{2}}$, which clearly gives rise to inaccuracies.

Heun

The *Heun method* [90] is an explicit method, which makes use of the forward Euler method as predictor and the trapezoidal rule as corrector. It can also be seen as a *2nd-order Runge-Kutta method* (RK2).

Discretizing (1.2) results in the following implementation scheme:

$$\begin{aligned} \tilde{v}_{n+1} &= v_n + h\frac{f_n}{m} \ , \quad \text{predictor,} \\ x_{n+1} &= x_n + \frac{h}{2}(v_n + \tilde{v}_{n+1}) \ , \\ f_{n+1} &= f\left(x_{n+1}, \tilde{v}_{n+1}\right) \ , \\ v_{n+1} &= v_n + \frac{h}{2}\frac{f_n + f_{n+1}}{m} \ , \quad \text{corrector.} \end{aligned} \tag{1.26}$$

Again, it should be noted that both x_{n+1} and f_{n+1} only depend on the predicted velocity $\tilde{v}_{n+1}$, and this gives rise to inaccuracies.

4th-order Runge-Kutta (RK4)

The *4th-order Runge-Kutta* method (RK4) [90] is an explicit iterative method which is widely used to solve ODEs with improved accuracy.

Discretizing (1.2) results in the following implementation scheme:

$$\begin{aligned} x_{n+1} &= x_n + \frac{1}{6}(l_1 + 2l_2 + 2l_3 + l_4) \ , \\ v_{n+1} &= v_n + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4) \end{aligned} \tag{1.27a}$$

where:

$$\begin{aligned} l_1 &= hv_n \ , \quad l_2 = h(v_n + \frac{k_1}{2}) \ , \\ l_3 &= h(v_n + \frac{k_2}{2}) \ , \quad l_4 = h(v_n + k_3) \ , \\ k_1 &= h\frac{f_n}{m} \ , \quad k_2 = h\frac{f(x_n + \frac{l_1}{2}, v_n + \frac{k_1}{2})}{m} \ , \\ k_3 &= h\frac{f(x_n + \frac{l_2}{2}, v_n + \frac{k_2}{2})}{m} \ , \quad k_4 = h\frac{f(x_n + l_3, v_n + k_3)}{m} \ . \end{aligned} \tag{1.27b}$$

It should be noted that, for each sample, both the velocity and the nonlinear force of (1.1) need to be evaluated four times, therefore weighing on the efficiency of the simulation (see Section 1.4.3).

1.3.2 Experimental results

In order to evaluate the chosen numerical methods, it is useful to compare the behavior of the corresponding simulations against the analytical results provided in Section 1.2.1.

The main references used to quantitatively assess the reliability of a particular numerical method *during* contact, are equations (1.3) and (1.21), which express respectively the compression x and the Hamiltonian H of the system as functions of the compression velocity v. Considering (1.3) and defining $x_{\max}$ as in (1.4), the percentage error on x is calculated by means of the following formula:

$$\%err_x = 100 \cdot \left| \frac{\max_n \{x_n - x(v_n)\}}{x_{\max}} \right|. \tag{1.28}$$

Defining H^{sim} as the Hamiltonian of a particular numerical simulation, ΔH_τ as in (1.19), and recalling (1.21), the percentage error on H is calculated by means of the following expression:

$$\%err_H = 100 \cdot \left| \frac{\max_n \left\{H^{\text{sim}}(v_n) - H(v_n)\right\}}{\Delta H_\tau} \right|. \tag{1.29}$$

Equations (1.28) and (1.29) should be seen more properly as measures of the *maximum deviation* of the discrete-time versions of x and H, respectively from the analytical curves $x(v)$ and $H(v)$, with respect to the entire variation range of the quantity considered (that is, $x_{\max}$ or ΔH_τ).

Another indicator which allows to evaluate the accuracy and consistency of the numerical methods at release from contact is provided by the value of the output velocity v_{out} computed numerically as a zero of (1.6).

Throughout the following example simulations some values of parameters are kept constant: $m = 10^{-2}$ Kg, and $F_s = 44.1$ kHz (that is, a standard audio sample rate).

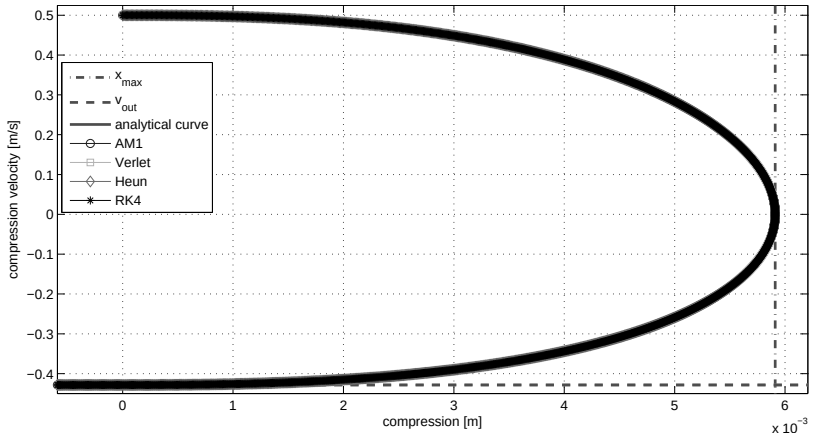
Reference simulations

In order to verify the simulations and provide a reference, the parameters of the model are set to a "safe" configuration, that is to far-from-extreme values. In this case, contact extends over many samples, thus ensuring that the numerical simulations are influenced only to a negligible extent by the chosen sampling rate (see Section 1.1.3), and should more likely behave as the original continuous-time system.

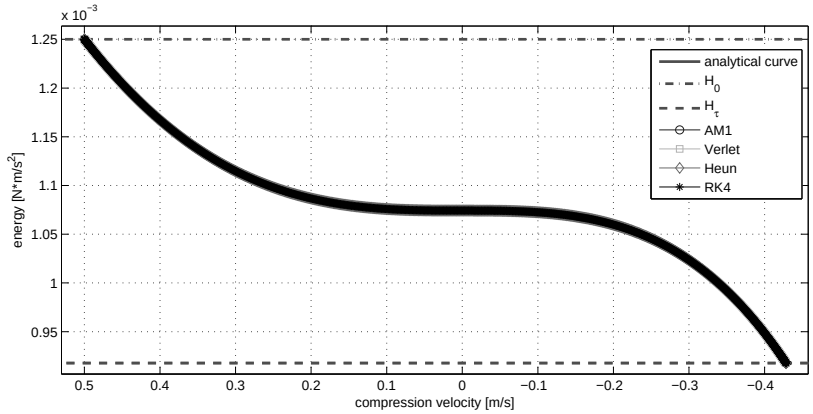
This is confirmed qualitatively by Fig. 1.6, where the plots of all the simulations substantially overlap and coincide with the analytical curve. Moreover, Table 1.2 offers a quantitative evaluation of the simulations, showing the errors introduced by the different numerical methods considered.

Critical regions of parameters

It has been found empirically that when $\tau^{\text{sim}} \leq 4$ samples the errors on x, H and v_{out} heavily increase, resulting in an extremely poor reliability of any numerical simulation. The obvious reason for this behavior is that, since only very few samples of data are available, the numerical systems are totally unable to describe the



(a) Phase portraits. The two tangent lines represent the maximum compression $x_{\max}$ calculated by (1.4) and the output velocity v_{out} computed numerically as a zero of (1.6).



(b) Energy behaviors. The two horizontal lines display $H_0 = T_0$ and $H_\tau = T_\tau$, that is respectively the initial and the final Hamiltonian.

Fig. 1.6. Qualitative comparison of different methods in reference simulations. The values of parameters are $k = 10^3$ N/m$^\alpha$, $\mu = 0.5$ s/m, $\alpha = 1.5$, $v_{\text{in}} = 0.5$ m/s.

Table 1.2. Summary of percentage errors in reference simulations. The values of parameters are $k = 10^3$ N/m$^\alpha$, $\mu = 0.5$ s/m, $\alpha = 1.5$, $v_{\text{in}} = 0.5$ m/s. The errors on x and H are calculated according to (1.28) and (1.29) respectively, while the output velocities of the simulations are compared to a value computed numerically as a zero of (1.6).

%err	AM1	Verlet	Heun	RK4
x	0.255	0.018	0.319	0.005
v_{out}	$+2 \cdot 10^{-5}$	$+2 \cdot 10^{-6}$	$-3 \cdot 10^{-5}$	$+2 \cdot 10^{-6}$
H	$3 \cdot 10^{-4}$	0.052	$4 \cdot 10^{-4}$	$1 \cdot 10^{-5}$

original continuous-time counterpart. Hence, in the study hereafter, only values of parameters resulting in $\tau^{\text{sim}} > 4$ samples are considered.

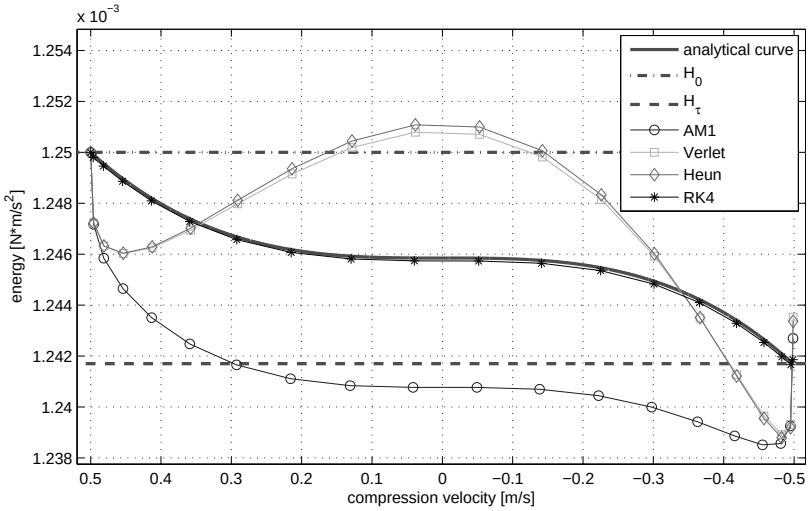


Fig. 1.7. Comparison of Hamiltonians for different implementations of a simulation example following *case 1*. The values of parameters are $k = 10^7$ N/m$^\alpha$, $\mu = 0.01$ s/m, $\alpha = 1.3$, $v_{\text{in}} = 0.5$ m/s. The contact time equals 19 samples.

It has been observed that when $\mu \to 0_+$ and/or as the contact time τ^{sim} decreases (that is, for "hard" impacts), the behavior of most numerical implementations tends to become inconsistent with the continuous-time system. Hereafter, the numerical systems are studied for these two critical configurations, respectively named *case 1* and *case 2*.

Case 1: low dissipation ($\mu \to 0_+$)

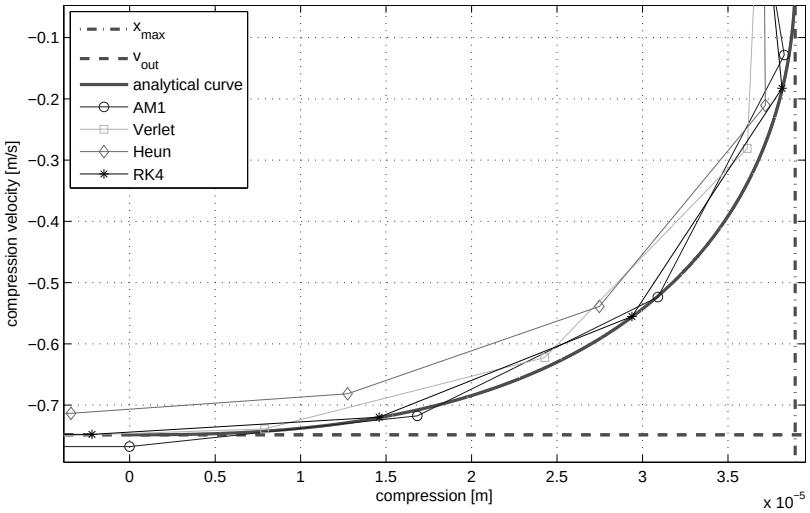
In case of low dissipation, the Hamiltonian of both Verlet- and Heun-discretized systems is prone to oscillations, while contact ends in an inconsistent final energy state: typically, $H_\tau^{\text{Verlet, Heun}} > H_\tau$, where H_τ is defined as in (1.18) and is calculated using values of v_{out} computed numerically as zeros of (1.6). As for AM1-discretized systems, these generally tend to dissipate too much energy during contact (that is, $H^{\text{AM1}} < H$), while slightly gaining spurious energy as the contact ends (that is, $H_\tau^{\text{AM1}} > H_\tau$). On the other hand, RK4-discretized systems generally behave quite consistently both during and after the contact interaction (that is, $H^{\text{RK4}} \approx H$).

Figure 1.7 shows the Hamiltonian of a simulation example with low dissipation ($\mu = 0.01$), while Table 1.3(a) shows the resulting percentage errors.

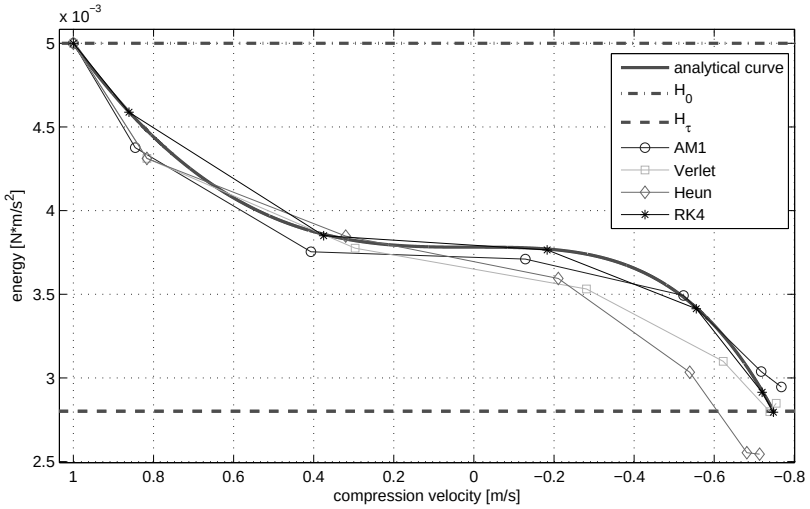
Case 2: hard impacts

With the exception of RK4-discretized systems, in this case the simulations usually result in more spread errors: x^{sim}, H^{sim} and $v_{\text{out}}^{\text{sim}}$ all tend to substantially deviate from the respective reference curves and values.

Figure 1.8 shows a hard impact simulation example following the values of parameters adopted in Fig. 1.1 for $v_{in} = 1$ m/s, while Table 1.3(b) shows the corresponding percentage errors. The resulting contact time τ equals 6 samples.



(a) Detail of phase portraits.



(b) Hamiltonians.

Fig. 1.8. Comparison of phase portraits and Hamiltonians for different implementations of a simulation example following *case 2*. The values of parameters are the same as in Fig. 1.1 with $v_{\text{in}} = 1$ m/s. The contact time equals 6 samples.

It is worth noticing that the RK4 method has been proved to behave quite consistently across disparate configurations of parameters. Therefore, a highly oversampled RK4-discretized system can be taken as a reference, being able to provide extremely accurate simulations.

Table 1.3. Percentage errors from example simulations of *case 1* and *case 2*. The last column shows the error yielded by the approximate value $\tilde{v}_{\text{out}}$ with respect to the value computed numerically: it is worth noticing that in both cases the error is lower than those yielded by the simulations.

(a) Simulation example following *case 1*. The values of parameters are the same as in Fig. 1.7.

%err	AM1	Verlet	Heun	RK4	$\tilde{v}_{\text{out}}$
x	1.011	1.083	1.136	0.052	-
v_{out}	+0.039	+0.073	+0.067	+0.006	$-1\cdot10^{-5}$
H	61.302	59.542	63.042	1.427	-

(b) Simulation example following *case 2*. The values of parameters are the same as in Fig. 1.1 for $v_{\text{in}} = 1$ m/s.

%err	AM1	Verlet	Heun	RK4	$\tilde{v}_{\text{out}}$
x	4.381	4.418	19.506	0.412	-
v_{out}	+2.551	+0.839	−4.692	−0.105	−0.013
H	7.885	9.475	23.387	0.410	-

1.4 Improved numerical simulations

1.4.1 Exploitation of analytical results

In this section some solutions are proposed that allow to fix the inconsistencies pointed out in Section 1.3.2. The aim is to improve the accuracy and reliability of simulations which make use of the impact model under study, in view of their implementation as real-time applications.

Maximum compression bound

One possible and very simple approach is to bound the compression [81], for which the following condition must hold:

$$x_n \leq x_{\max}, \quad \text{for all } n, \tag{1.30}$$

where $x_{\max}$ is a constant calculated by (1.4). Hence, the suggested solution is that of forcing x_n to its maximum allowable value $x_{\max}$ whenever this is exceeded. This also affects the Hamiltonian of the system by decreasing the elastic potential energy V_n.

As for the implementation of the suggested solution, the computation of $x_{\max}$ only needs to take place in correspondence to an impact event, and more precisely as soon as the impact velocity v_{in} is known.

Hybrid numerical-analytical computation

A more complex solution, alternative to the one described above, consists in computing the compression velocity v_n *numerically* – that is as a result of a numerical

simulation, using for example one of the numerical methods described in Section 1.3.1 – and employing it in (1.3) in order to calculate the corresponding value of compression $x(v_n)$ *analytically* [80]. Different numerical methods may need different implementations of this solution.[1]

As a result, the corrected numerical system strictly follows the analytical curves $x(v)$ and $H(v)$.

Since the computation is to be made for the whole duration of the contact interaction at each sample, this solution noticeably increases the computational load of the simulation.

Output velocity bound

The solutions above can be applied only while the contact interaction lasts, and therefore they cannot control the behavior of a numerical system upon the end of the impact. At that time, the energy content (that is, the residual kinetic energy $T_\tau^{\text{sim}} = H_\tau^{\text{sim}}$) can be controlled by forcing the output velocity to the approximate value $\tilde{v}_{\text{out}}$ [80, 81].

Considering that the error introduced by (1.8) depends solely on the product μv_{in} (see Fig. 1.3 and Table 1.1), it is advisable to apply the correction only when the product μv_{in} corresponds to an acceptable error, or the risk is to even worsen the behavior of the numerical system. However, the use of this conditional correction always implies a trade-off: on the one hand it guarantees not to introduce errors greater than a chosen maximum, while on the other hand 1) within the excluded range of μv_{in}, the output velocity is not controlled and therefore depends only on the plain numerical method chosen, 2) within the included intervals of μv_{in}, the risk is that the correction introduces errors even greater than those provided by the non-corrected numerical system (this is true especially for non-critical regions of parameters).

Once the output velocity has been forced to $\tilde{v}_{\text{out}}$, the corresponding compression should be set to 0, this way adhering to the compression-force characteristics shown in Fig. 1.2 by closing their numerical counterpart at $(x = 0, f = 0)$, and ensuring that the final potential energy V_τ^{sim} is set to zero.[2]

The computation of $\tilde{v}_{\text{out}}$ only needs to take place in correspondence to an impact event, for example as soon as the impact velocity v_{in} is known.

1.4.2 Numerical simulations with corrections

It was found that the *maximum compression bound* is only effective for some very hard impacts, where low-order numerical methods especially are not able to describe the compression curve accurately. In those cases, this cheap correction can be useful as it can contain the production of spurious potential energy related to

[1] For example, the forward Euler method [90] cannot be fully corrected, as the computation of the velocity v is completed before the computation of the force f (not being a predictor-corrector nor an iterative method). As as result, at the first sample the displacement x must be computed by the plain numerical method, this way introducing considerable errors, especially for hard impact configurations.

[2] However, in this way a small temporal error is generally introduced.

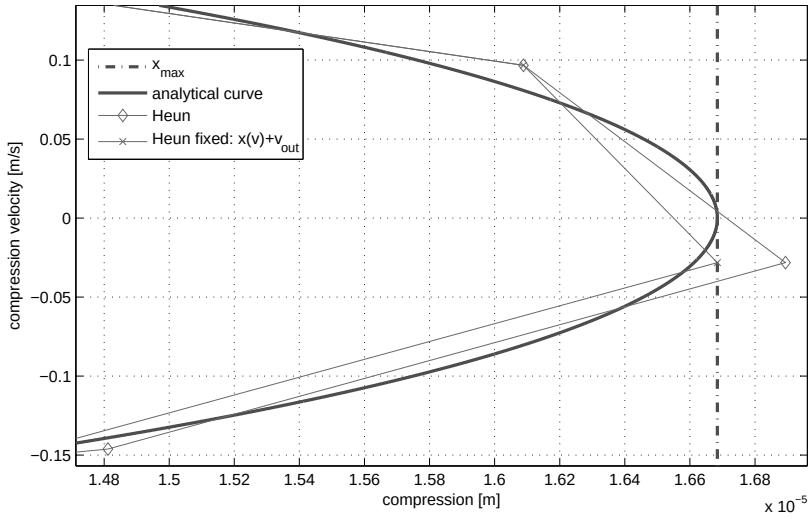


Fig. 1.9. Detail of phase portrait of Heun-based simulations with and without correction (the compression x is forced to $x_{\max}$ when exceeded). The values of parameters are $k = 10^7$ N/m$^\alpha$, $\mu = 0.1$ s/m, $\alpha = 1.1$, $v_{\text{in}} = 0.3$ m/s.

an inconsistently high compression. Figure 1.9 shows a simulation example where such correction actually improves the behavior of the numerical system.

Conversely, the other suggested corrections (namely, the *hybrid numerical-analytical computation* and *output velocity bound*) proved to be very effective and useful for most configurations of parameters and any numerical method. Therefore, in what follows only such corrections are taken into account. In order to test them, they were applied to the worst behaving simulation examples provided in Section 1.3.2: Fig. 1.10 and 1.11 show a comparison of Heun-based simulations following respectively *case 1* and *case 2*, with and without corrections.

Improved energy behavior

Equation (1.29) enables to quantitatively assess the improvements on the energy consistency of the numerical simulations.

When the hybrid correction described in Section 1.4.1 is applied, any simulation strictly adheres to the analytical curves $x(v)$ and $H(v)$ during contact, that is the respective percentage errors, as defined in (1.28) and (1.29), are equal to zero.[3]

As for the energy state upon the end of the interaction, the error on H_τ depends either on the error introduced by $\tilde{v}_{\text{out}}$ (only when the output velocity bound described in Section 1.4.1 is actually applied), or on the error introduced by the plain numerical simulation. As already stated in Section 1.4.1, in the first case the maximum error on v_{out} is predictable, and clearly the same goes for the error on H_τ.

[3] Not taking into account the inherent errors related to the representation of numbers in the digital domain.

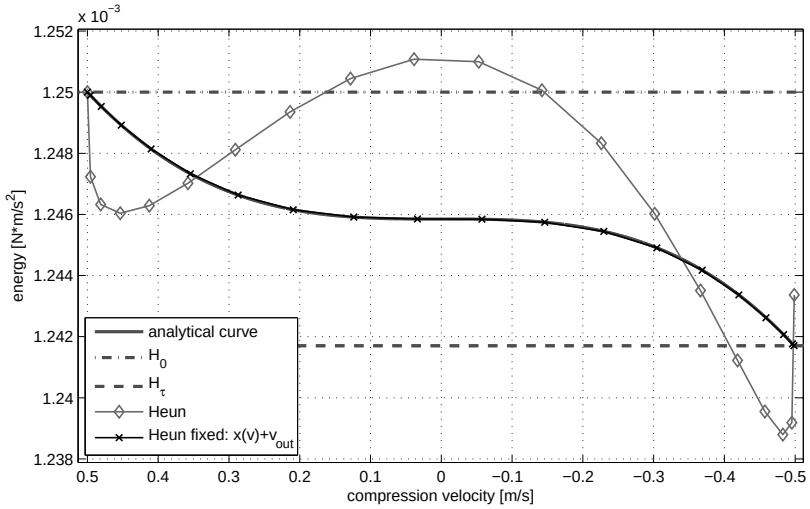


Fig. 1.10. Comparison of the Hamiltonians of a Heun-based simulation example following *case 1*, with and without corrections.

The trend of error on H resulting from simulations with and without corrections is depicted in Figures 1.12(a) and 1.12(b), which show that even the best numerical method among those considered (that is RK4) can be improved, especially for critical values of parameters (see Section 1.3.2) such as $\mu v_{\text{in}} \to 0_+$.

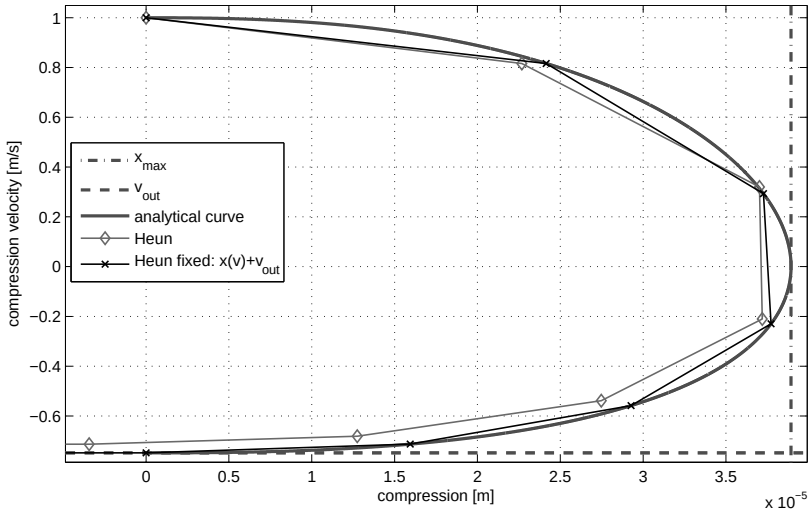
Sequence of impacts

In order to better appreciate the importance of the proposed corrections, a sequence of impacts has been implemented, where the input velocity for each subsequent collision is equal to the output velocity of the previous one: $v_{\text{in}}^i = v_{\text{out}}^{i-1}$, where i is the index of the i-th impact. This way the system simulates a sequence of rebounds, as if a conservative force[4] (e.g., gravity) was applied during free motion. Thanks to this setup it is possible to track the accumulation of energy anomalies at each contact interaction.

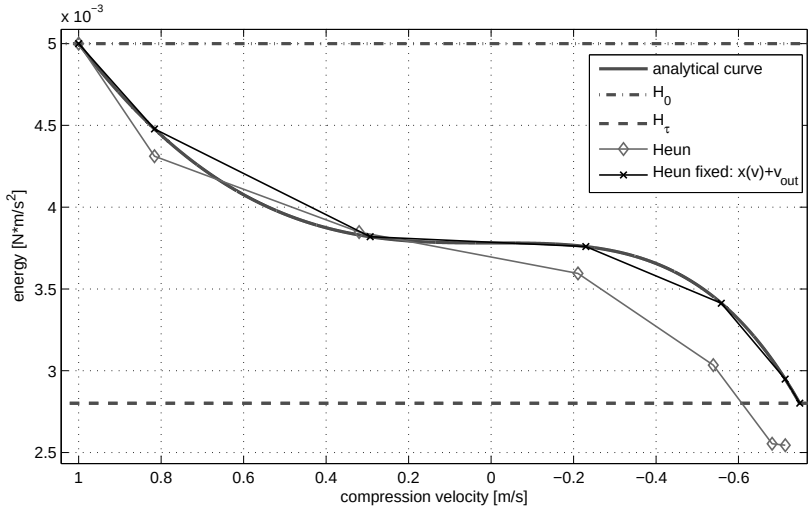
To this end, the residual energy $H_\tau^{i,\text{sim}}$ after the i-th impact of numerical simulations with and without corrections was examined and compared with the residual energy $H_\tau^{i,\text{ref}}$ of a 16× oversampled RK4-based simulation used as a reference. Also, the maximum deviation of H^{sim} with respect to the analytical curve $H(v)$ was measured according to (1.29) along the whole sequence of impacts.

Table 1.4 shows the errors on the residual energy $H_\tau^{100,\text{sim}}$ and the maximum deviation of H^{sim} occurred during a sequence of 100 impacts, for simulation examples following *case 1* and *case 2* with and without corrections. Notice that, since in some simulations the energy can strongly oscillate during contact (see for

[4] As shown in Section 1.2.2, when an external force is applied to the system, no closed-form analytical result is available, this way making the corrections provided in Section 1.4.1 unsuitable.



(a) Phase portraits.

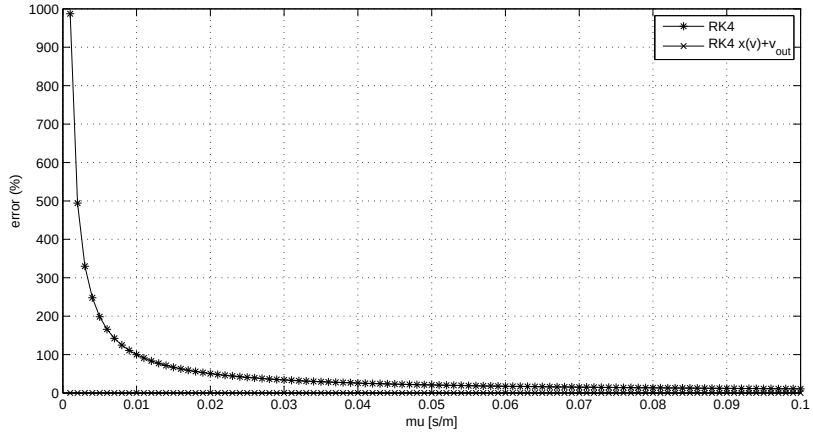


(b) Hamiltonians.

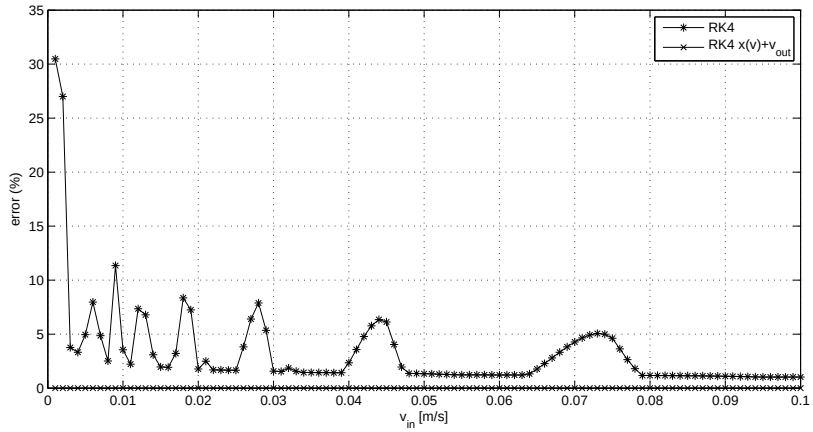
Fig. 1.11. Comparison of a Heun-based simulation example following *case 2*, with and without corrections.

example Fig. 1.7), the maximum deviations from $H(v)$ do not necessarily reflect the final (accumulated) errors.

From the results shown in Table 1.4 it is self evident that – even in case of errors apparently negligible for a single impact (recall the errors relative to RK4 reported in Table 1.3) – indeed, for multiple impacts, energy inconsistencies accumulate and can become noticeable – if not disastrous – after a certain number of contact interactions.



(a) Percentage error on H as μ varies.



(b) Percentage error on H as v_{in} varies.

Fig. 1.12. Trend of error on the Hamiltonian for RK4-based simulations with and without corrections, for small values of μ and v_{in}. For the values of μv_{in} considered, the output velocity bound is always applied (see Table 1.1). The values of parameters, where kept constant, are $k = 10^9$ N/m$^\alpha$, $\mu = 0.5$ s/m, $\alpha = 1.5$, $v_{\text{in}} = 0.5$ m/s. At each corresponding point of the two parallel plots, the product μv_{in} is the same. It can be noted that, when not using any corrections, the weight of μ on the error is clearly greater than that of v_{in}.

1.4.3 Computational cost

In this section, the computational costs of both the numerical implementations seen in Section 1.3.1 and the corrections suggested in Section 1.4.1 are taken into account. In particular, the cost has been measured as the number of operations (namely, memory write/read accesses and arithmetical operations) needed to execute an algorithm.

Table 1.5(a) summarizes the cost of the algorithms implementing the numerical methods considered so far. Since the AM1-based implementation makes use of

Table 1.4. Summary of errors on the residual energy $H_\tau^{100,\mathrm{sim}}$ and maximum deviations of H^{sim}, occurred during a sequence of 100 impacts. The last columns show the errors resulting from simulations corrected using the hybrid computation and output velocity bound described in Section 1.4.1. The error threshold for the output velocity bound is set to 0.1%, and therefore the correction is always applied (see Table 1.1). It is worth noticing that the maximum deviation from $H(v)$ of the 16× oversampled RK4 simulation is 0.021% for Table 1.4(a) (that is, greater than the deviation of the corrected simulations), and 0.027% for Table 1.4(b) (that is, very close to the deviation of the corrected simulations).

(a) Simulation examples following *case 1*. The values of parameters are the same as in Fig. 1.7, except for input velocities decreasing at each impact event.

%err H	AM1	Verlet	Heun	RK4	corrected sim.
accumulated err.	12.023	10.061	147.040	0.906	0.001
max. deviation	72.958	72.106	63.040	14.461	0.009

(b) Simulation examples following *case 2*. The values of parameters are the same as in Fig. 1.1, starting with $v_{\mathrm{in}} = 1$ m/s and for input velocities decreasing at each impact event.

%err H	AM1	Verlet	Heun	RK4	corrected sim.
accumulated err.	24.287	15.782	946.616	2.151	0.003
max. deviation	69.149	43.959	27.418	6.250	0.032

Newton's method, its cost is displayed on two sub-columns: the first one shows the constant cost per sample, while the second one (in italics) shows the cost of a single iteration of Newton's method. Notice that the number of iterations per sample is not predictable.

Table 1.5(b) shows the cost of the corrections considered. For comparison, the last column reports the cost of a single zero-finding iteration on (1.6). The number of iterations depends on μ and v_{in}, and is usually in the order of some tens.[5] It is clear that, despite being more precise than the approximate value $\tilde{v}_{\mathrm{out}}$, the value computed numerically as a zero of (1.6) implies several times the number of operations required by $\tilde{v}_{\mathrm{out}}$. Moreover, one could use a series expansion of higher order for $\tilde{v}_{\mathrm{out}}$ (so far, the 4th-order expansion of (1.9) has been used), thus increasing the accuracy of the approximation, while marginally affecting the computational cost (recall that such calculation only needs to take place once in correspondence of an impact event).

Since the computational load of simple write/read operations is generally low (if not negligible), two totals for each column are reported: one excluding write/read operations and, in brackets, one accounting for them.

Recalling that the hybrid correction only affects the computational cost during contact, whereas the output velocity bound is applied at most[6] once per impact event, from Table 1.5 one can infer that Verlet- or Heun-based simulations with corrections are roughly three times as efficient as plain RK4-based simulations

[5] The number of iterations was empirically found as being usually between 15 and 40. Moreover, as μ decreases, the number of iterations increases.

[6] As shown in Section 1.4.1, the output velocity bound is applied conditionally.

Table 1.5. Number of operations needed by the numerical methods shown in Section 1.3.1 and the corrections described in Section 1.4.1. The totals in brackets account for write/read operations.

(a) Since the AM1-based implementation makes use of Newton's method, its cost is displayed on two subcolumns: the first one shows the cost per sample, while the second one (in italics) shows the cost of a single Newton's method iteration.

	AM1		Verlet	Heun	RK4
write	6	*10*	5	5	20
read	16	*33*	20	21	72
$+/-$	3	*8*	7	8	22
$\times$	8	*11*	5	5	18
$\div$	0	*1*	3	3	8
bit-shift	0	*0*	2	3	6
exp	0	*1*	1	1	4
log	0	*0*	0	0	0
compare	0	*4*	1	1	4
TOTAL	11 (33)	*25 (68)*	19 (44)	21 (47)	62 (154)

(b) The last column (in italics) shows the cost of a single zero-finding iteration on (1.6).

	hybrid correct.	$\tilde{v}_{\text{out}}$ bound	v_{out} as zero of (1.6)
write	1	2	*1*
read	7	4	*3*
$+/-$	6	5	*4*
$\times$	7	6	*3*
$\div$	3	4	*1*
bit-shift	0	1	*0*
exp	0	1	*0*
log	1	0	*1*
compare	1	1	*0*
TOTAL	18 (26)	18 (24)	*9 (12)*

during free motion, and almost twice as efficient during contact. On the other hand, while the exact computational load of an AM1-based simulation is not predictable, it can be noted that it already matches the cost of a RK4-based one after two iterations of Newton's method.

1.4.4 Evaluation of methods

The different implementations considered can be eventually evaluated in the light of the results regarding their accuracy (see Sections 1.3.2 and 1.4.2) and computational load (see Section 1.4.3).

It can be stated that, among the non-corrected implementations, the best all-round performance is achieved by RK4-discretized systems, although they are

quite computationally expensive. On the other hand, a corrected Verlet-discretized system is generally approximately as good as a non-corrected RK4 implementation, at a fraction of its computational load.

As for AM1-based implementations, it was shown that they behave quite poorly in the critical regions identified as *case 1* and *case 2*. These poor results in terms of accuracy, together with a generally high and moreover non-predictable computational load, set the AM1 method as a hardly recommendable choice. Apparently such conclusion can be quite surprising, especially if one considers that AM1 is the only implicit and A-stable method among those considered. However, the presence of nonlinearities, together with the inaccuracies introduced by the K-method and Newton's method, justify the behavior of AM1-based implementations.

2

Tools for ecologically-founded sound synthesis and design

In this chapter, first some existing sound synthesis tools are briefly reviewed, which are suitable for applications in interactive sonification [56] and sonic interaction design [98]. Thereafter, a recently developed open-source software package for ecologically-founded sound synthesis and design is described in detail. This package comprises a library of sound synthesis algorithms, examples and advanced tools useful in interactive contexts.

2.1 Existing sound synthesis tools for interactive sonification

Drawing inspiration from studies on ecological acoustics, everyday-sound perception and categorization [50] (see the introduction to this thesis), a plethora of sound synthesis tools have been produced [49, 110, 111]. Some notable examples are:

- An example of physically-informed/-inspired algorithms which are able to synthesize everyday and other non-musical sounds, is provided by Perry Cook's "Physically Inspired Sonic Modeling" (PhISM) [18] (included in the C++ library STK [21]), which consists of two different typologies of synthesis algorithms implementing respectively spectral additive modeling (PhISAM), and stochastic event modeling (PhISEM). For example, PhISM is capable of synthesizing hand claps [86], the sound of footsteps [19], ice cubes in a glass and rain drops [20].
- As one of the outcomes of the EU-funded project *SOb*, which is briefly reviewed in the introduction to this thesis, a software package was developed, running in *pure data*[1] (Pd) and implementing several sound models. The models are divided in two categories: low-level physically-based models – namely, modeling impact [6] and friction [10] interactions – and high-level physically-informed/-inspired models which make use of the low-level ones, this way obtaining for example bouncing, breaking, crumpling and rolling sounds. As notable application examples, it is worth mentioning that the crumpling model was exploited [41] for synthesizing the sound of crushing and of footsteps on aggregate

[1] An open source graphical programming environment, especially dedicated to sound processing and production (`http://puredata.info/`).

grounds as snow or gravel, while in [94] it is explained how continuous sonic feedback provided by the rolling sound model can be used in interaction design.
- Other recently introduced tools which allow to synthesize everyday sounds are *xoxos*' VST plugins "Sounds of Nature"[2], and Andy Farnell's Pd patches[3].

2.2 The Sound Design Toolkit

The recent flourish of researches involving sound design and sonification in interactive contexts (as, for example, human-computer interfaces and video games) share the observation that sound creation still suffers from a lack of well defined paradigms and methodologies. In particular, this shortage can be ascribed to the fact that sound utilization in such contexts exhibits unique and original requirements: for example, sound should be dynamic, informative, functional to the interaction or evoke *ad hoc* emotional responses. To this end sonification tools and systems should offer extensive control over the sound design process, sound generation and continuous real-time manipulation [58]. On the other hand, if compared with their auditory equivalent, current techniques in computer graphics [73,74,107] offer highly dynamic, controllable and effective visual representations of objects, interactions and environments.

Current sound creation methods still largely rely on physical production (Foley art[4]) and sampling techniques. Indeed, the physical production of sound in the Foley tradition ensures both high correlation with reality and great expressivity. However, as a matter of fact, such physical production is highly expensive, entailing several hours of recording sessions with both Foley artists and recording engineers. Furthermore, sounds produced in this way can only be manipulated to some extent once fixed in recorded samples. Also, sample-based processing requires a sort of sculptural, artistic attitude.

In contrast with such practices, modern sound synthesis techniques – with their inherent parametric control – are able to handle continuous sound feedback in interactive contexts more dynamically and effectively, and this is especially true for physically-based sound synthesis. Nonetheless, with such sound synthesis techniques the design of expressive sound feedback is anything but straightforward, since they require quite complex control methodologies. Moreover, tools for sound design in contexts where functional sound feedback is required are still lacking, while most of the currently available tools are borrowed and readapted from musical production,[5] and therefore usually embrace an aesthetic (as opposed to functional) approach.

[2] `http://www.xoxos.net/`
[3] `http://www.obiwannabe.co.uk/`
[4] `http://en.wikipedia.org/wiki/Foley_artist`
[5] On the other hand, digital sound synthesis historically has strong foundations in music production. See the "Sound and Music Computing" (*SMC*) network's website for further information: `http://smcnetwork.org/`.

2.2.1 Introduction

The Sound Design Toolkit (SDT) is a software product whose development started within the EU-funded project *CLOSED* (which is reviewed in the introduction to this thesis), and is made up of a set of physically-consistent tools for designing, synthesizing and manipulating ecological sounds in real-time. The aim of the SDT is to provide efficient and effective instruments for interactive sonification and sonic interaction design.

The SDT originates from the *SOb* library of sound models, which was implemented as a collection of patches and *externals* for Pd (see Section 2.1). Among other things, the SDT also offers patches and externals for Max/MSP[6], which is in a sense the advanced, yet commercial, counterpart to Pd.

In Pd's and Max/MSP's terminology, an *external* is a dynamic library (actually, a binary file) which provides some kind of processing, and corresponds to one ore more objects for use in their graphical programming environments. An *external*can have one or more inlets and outlets, respectively allowing to take input and provide output data. The are two different types of *externals*: control and signal ones. The first ones can handle (that is, take as input) and provide (that is, output) control messages only, while the second ones can either take and output signals or control messages. It is useful to notice that in Pd and Max/MSP the distinction between *signals* and *controls* basically refers to their update rate and synchronous or asynchronous nature. In detail, both environments provide two different schedulers (having varying degrees of mutual synchronicity) which handle respectively signals and controls at runtime: the first ones are treated as synchronous, audio-rate signals (yet, they do not need to be actual audio signals), while the second ones are handled as low-rate asynchronous messages. As for Pd and Max/MSP patches, they can either correspond to functions or self-contained programs, and can make use of one or more *externals*, together with, for example, arithmetical operators, digital filters, sliders or other GUI elements natively provided by the environment.

In particular, each SDT's *external* represents a physically-based or -informed/-inspired algorithm for sound synthesis (signal *externals*) or control (control *externals*). As for the latter, the generated control messages are meant to be used as controls for other sound synthesis processes. Finally, the SDT's patches make use of those *externals* in order to implement fully functional physically-consistent control and sound models. Also, they provide facilities for driving control parameters (for example, a graphical interface) and allow to route audio output through a computer's audio interface.

2.2.2 The software package

Development and distribution

In the first place, at the beginning of the development process the whole *SOb* library of sound models has been thoroughly revised, and the *externals* have been

[6] `http://www.cycling74.com/`

rewritten to comply with the programming layer *flext*[7]. Thanks to *flext*'s build system, the resulting source code can be compiled under different operating systems, for both Pd (MacOSX, Windows and Linux) and Max/MSP (MacOSX and Windows only).

The addition of support for Max/MSP offered the occasion to restyle the original *SOb* Pd patches, exploiting the enhanced interface and expanded features provided by the Max/MSP environment, this way offering more accessible and usable interfaces, and on-line documentation. Indeed, in addition to the standard help system – also present in Pd – which, for each *external*, invokes a tutorial patch describing its functionalities, Max/MSP features a way to display information about the inlets and outlets of any graphical object, directly from the editing window. In this way, the user can quickly review the functionalities of an *external*, understanding what this can process as input and provide as output. Moreover, the free runtime version of Max/MSP still allows users not owning the full-version software to use the SDT, essentially just loosing the possibility of writing new patches and of editing the ones provided.

Secondly, several new models have been added to the library, together with a growing collection of advanced examples which show the full potential of the models. Also, more recently, an advanced front-end user interface has been added, which provides unified access to the models and adds powerful features like sequencing, MIDI[8] and OSC[9] control, mixing and presets management. Finally, a detailed user manual[10] has been written, taking inspiration from the original Max/MSP documentation, which shows the package content and functionality, with particular attention to the description and operation of the *externals* provided.

The software package is open source and distributed under the Gnu General Public License (GPL), while the manual is distributed under the Gnu Free Documentation License (FDL).

In order to allow collaborative development and easy distribution, a SVN[11] repository has been set up,[12] which is publicly accessible and is currently being used to maintain the software. The SDT's repository is arranged according to a standard SVN structure: the directory `/trunk` contains the main line of development (at the moment set as private), under `/branches/SObWay` is the development line currently made available, while under `/tags` are the official releases, corresponding to major revisions in the history of the repository.

---

[7] `http://puredata.info/Members/thomas/flext`

[8] Musical Instrument Digital Interface: a protocol used to transmit music-related control messages. Despite its age, MIDI is still the most widespread protocol in musical applications.

[9] Open Sound Control: a control protocol which adds several improvements to MIDI (`http://opensoundcontrol.org/`).

[10] At the time of writing, the latest of version of the manual is included as part of the public deliverable 2.3 of the project *CLOSED*: `http://closed.ircam.fr/deliverables.html`.

[11] Subversion: an open source version control system.

[12] The SDT is available from the following SVN repository: `https://svn.sme-ccppd.org/svn/sobs/SoundDesignTools/`

Content

The SDT is split into three main sections, corresponding to separate subdirectories: the source code is provided under `/SDT_sources`, the patches can be found under `/SDT_patches`, while pre-compiled *externals* for MacOSX,[13] Windows and Linux (Pd only) are located under `/SDT_binaries`.

The main source files, each corresponding to a different *external*, are coded in C++ using *flext*, while some other auxiliary files are coded in C.

Patches for Pd and Max/MSP are provided in separate directories (respectively named `/for_pd` and `/for_Max`), which are further divided in a hierarchical fashion: at the lowest level, under `/help`, are the *help patches*, each of which is a thorough example exploring the features of a certain *external* and showing how it works; on a higher level, under `/examples`, are the *example patches*, which implement advanced utilizations of the models, for example introducing a high level control layer; at the highest level, under `/main_interface` (currently only available for Max/MSP), is the *front-end interface* mentioned in Section 2.2.2.

2.2.3 A library of everyday sounds

Drawing inspiration from Risset's catalog of computer-synthesized sounds [62] and Gaver's taxonomy of everyday sound [50], a catalog of contact sounds models [92] was produced within the project *SOb*. An expanded – yet not exhaustive – taxonomy of everyday sounds, taking also into account phenomena related to liquids and gases, has been further developed during the project *CLOSED* [59]. One of the aims of this new taxonomy was to set a landmark for the development of everyday-sound synthesis algorithms, in order to progressively cover a rich collection of everyday acoustic phenomena especially relevant in interactive contexts. The taxonomy is depicted in Fig. 2.1, which shows how various acoustic phenomena are linked with sound models that can be used to obtain them: that is, a hierarchy is established from low-level sound events to more complex, patterned or compound processes. In detail, basic sound models are presented at the bottom of the graph, while the second level shows simple sound events and textures straightly derivable from them. Processes that can be related to temporal patterns of single events and textures are presented at the third level. Lastly, the top level contains several examples of implemented simulations, while dashed connections represent expected dependencies for the simulations yet to be developed.

As an example, models of impact and friction have been exploited as basic acoustic events underlying complex acoustic phenomena: rolling, bouncing, breaking and crumpling sounds are modeled through elaborate temporal patterns on top of the impact model, while the friction model is used to simulate rubbing, squeaking and braking sounds.

Also, several new models have been added to the SDT's library:

- an enhanced crumpling model;
- a continuous-crumpling model;

[13] The *externals* for MacOSX are in universal binary format, that is they are compatible with both PPC and Intel Macs.

- various liquid-related sounds – as boiling, dripping, pouring and frying – have been implemented making use of a simple model of a collapsing bubble;
- as for gas-related sounds, a vacuum cleaner simulation have been implemented;
- a soft impact model.

More details on the new models are given in the following sections.

2.2.4 Low-level models

The low-level models directly simulate basic acoustic phenomena which can be regarded as the most basic elements in the SDT, and which are also used to render complex sound events, texture and processes (see Section 2.2.5). They include basic contact interactions between solids and the model of a liquid bubble collapsing.

Solids

The algorithms underlying solids sound models share a common structure which is shown in Fig. 2.2: two objects interact through what is called an *interactor*, which models the actual contact interaction (for instance, impact or friction). The resonators and the interactor are linked through feedback connections: in more detail, at each discrete-time instant (sample) both the objects send their internal states (namely, displacement and velocity at the interaction point) to the interactor, which in turn sends the newly computed interaction forces back to the objects, thus exciting them. Knowing the new applied forces, the objects are able to compute their new states for the next time instant, and so on.

Such feedback scheme distinguishes remarkably the SDT's models from other approaches to physically-based sound synthesis found in most existing implementations [72] and literature [23,103]. As a matter of fact, mainstream models usually adopt a feed-forward structure: a "exciter $\rightarrow$ resonator" configuration is adopted, where non-linearities are lumped in the interaction component (which, using musical examples, can represent the blow for wind instruments, a piano hammer, etc.). On the contrary, the low-level solids models of the SDT implement a feedback network according to the structure "object1 $\leftrightarrow$ interactor $\leftrightarrow$ object2", thus allowing to accurately model complex interactions and to output sound as the vibrations of both the interacting objects (for example, simulating two cups colliding). Also, the continuous interaction modalities provided by the SDT are memory consistent, that is the system takes record of each previous state during the interaction.

The non-linear characteristics of the interactors result in richness and dynamic quality of the sounds synthesized, even when using very basic resonators (that is, having few modes of resonance).

The SDT implements a modular approach, where any couple of solid objects and any contact interaction model can be combined in whatever configuration, thus offering extremely wide sonic possibilities.

Object models

Three distinct object models are provided:

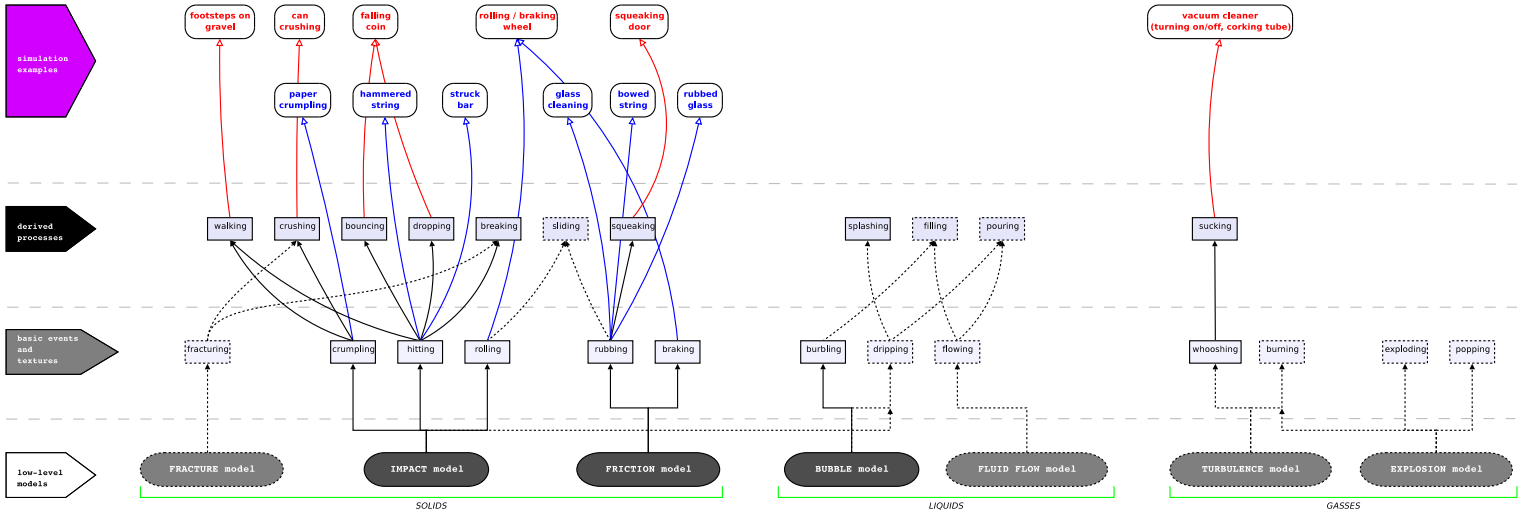


Fig. 2.1. Overview of a proposed taxonomy of everyday sounds with parent sound models. The graph develops bottom-up starting with low-level sound models. The second level contains basic events and sound textures founded on the low-level models. At the third level are processes based on one or more basic events and textures. Elements outlined by dashed lines are proposed and not yet implemented, while dashed connections represent expected dependencies.

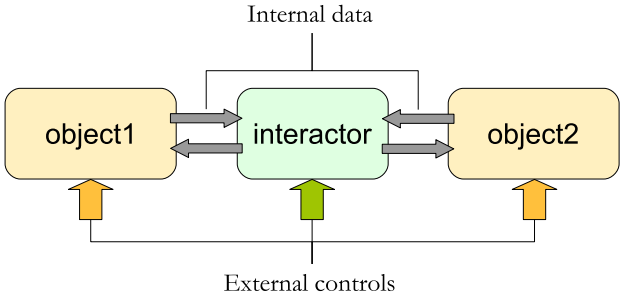


Fig. 2.2. The common structure underlying the SDT's algorithms simulating solid contacts.

Modal object - The modal paradigm [3, 112] supports particularly well the SDT's design approach for its physical generality and, at the same time, for its intuitive acoustic meaning. In the modal description a resonating object is described as a system of a finite number of parallel *mass-spring-damper* structures. Each mass-spring-damper structure models a damped mechanical oscillator which represents a normal *mode of resonance* of the object. The oscillation period, mass and damping coefficient of each oscillator correspond respectively to the *frequency of resonance*, *gain* and *decay time* of each mode. Section A.1 of Appendix A offers a detailed review of the modal resonator.
In the SDT implementation it is possible to choose the actual number of modes and to control their properties individually. Furthermore, each modal object can have a number of *pickup points* from which the sound is output. There must be at least one pickup point, and the first one is also where contact takes place, that is it is the *interaction point*.

Inertial object - It simulates a simple point mass. Obviously this kind of objects are useful solely to excite other resonators.
The only settable property is the *mass* of the object.

Waveguide object - The digital waveguide technique (DWG) [102] models either the propagation of waves along elastic media or air pressure inside, for example, a tube. Waveguide synthesis is widely exploited in the computer music community, where it has been mainly used for the faithful simulation of existing musical instruments. In the one-dimensional case implemented for the SDT [78, 79], the waveguide object models an ideal elastic string with propagation losses.
It is possible to set the *length*, *tension* and *mass* of the string. Furthermore, one can set the position of the *interaction point* along the length of the string, which also coincides with the only available pickup point (that is, the place from where the sound is output).

All the object models share the same interface, giving access to their internal state described by the displacement and velocity at the interaction point.

Contact interaction models

Most of the contact interactions between solids can be reduced to impulsive contacts (for example, impacts) and continuous contacts (like friction, rolling, scraping, etc.). In the SDT's framework an interactor corresponds to a contact model, representing the local interaction between two contacting objects. The forces resulting from the contact are calculated and then provided to the objects, inducing vibrations which can be output as sound.

Two implementations of impact models (one of which is a linearized version) and one friction model are provided:

Impact interactor - It implements a non-linear impact force model [61]. It takes the overall compression – that is, the difference of displacements of the two contacting objects at the interaction point – as input, and returns the resulting impact force.
The model is described by a contact force which is the sum of an *elastic* and a *dissipative* component. The elastic component is parameterized by a *stiffness* coefficient and an *exponent* of non-linearity which depends on the local geometry around the contact area, while the dissipative component is parameterized by a *dissipation* coefficient.

Linear impact interactor - implements a linear impact force. As it is a linearized version of the impact interactor described above, the non-linear exponent is set to 1, while the *stiffness* and *dissipation* coefficients are still considered. The adoption of this linear model is motivated by the improved computational efficiency of the resulting numerical system.

Friction interactor - implements a non-linear friction force [10]. It takes the relative velocity of the two rubbing objects as input, and returns the friction force computed.
The model is described by a friction force, which is made up of three components representing *elasticity*, *dissipation* and *viscosity*, in addition to a pseudo-random component describing the noise related to the *surface roughness*. The model is parametrized by some others quantities: *dynamic friction* and *static friction* coefficients, *normal force* between the two rubbing objects, a *breakaway* coefficient and the *Stribeck velocity*.

An in-depth study of the impact model is provided in Chapter 1, while a brief overview of the friction model is provided hereafter. The friction model describes the average behavior of a multitude of micro-contacts made by hypothetical bristles extending from each of two sliding surfaces. The main equations modeling the friction interaction are:

$$\begin{cases} \dot{z}(z,v) = v(t)\left[1 - \alpha(z,v)\frac{z(t)}{z_{ss}(v)}\right] \\ \\ f(z,\dot{z},v,w) = \sigma_0 z(t) + \sigma_1 \dot{z}(t) + \sigma_2 v(t) + \sigma_3 w(t) \end{cases} \tag{2.1}$$

where v represents the *relative velocity* between the two rubbing objects, z is the *mean bristle displacement* and w is a generic pseudo-random function which introduces a *noise component*. The functions α and z_{ss} have been described according to expressions previously proposed in [36].

Figures 2.3 and 2.4 show two help patches for Max/MSP, implementing respectively impact and friction interactions between two solid objects.

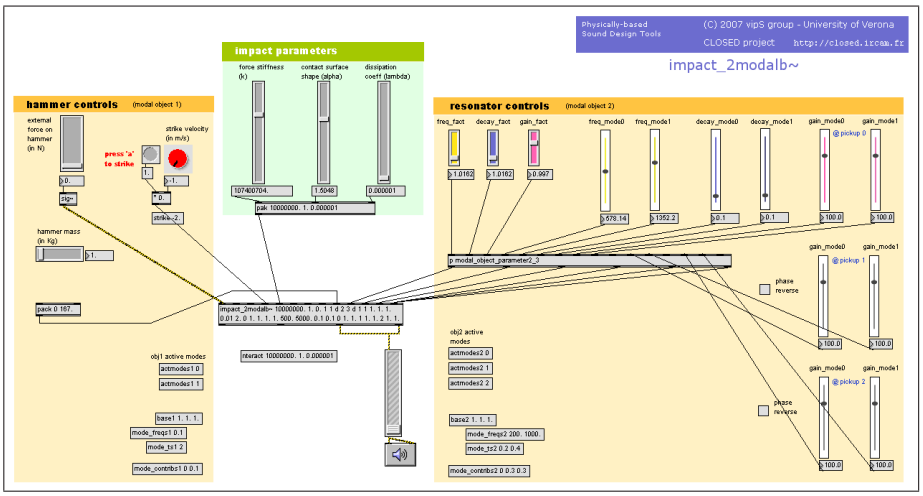


Fig. 2.3. A Max/MSP patch implementing the contact interaction between two solid objects. The object in the center, which is named impact_2modalb~ and takes several inputs from sliders or other control devices, is the graphical representation of the corresponding homonym *external* implementing the impact between two modal resonators.

Liquids

Basic events responsible for acoustic emission in liquids have been taken into account [34], resulting in a low-level model which simulates the formation of single resonating cavities (bubbles), as observed in dripping, boiling, or pouring.

Bubble model

The formation of radially oscillating bubbles under the surface of a volume of liquid is modeled assuming that the impulse response of a single bubble is well represented by a damped sinusoid with time-varying frequency [110]. In particular, frequency increases as the radius of a bubble decreases in time.

One of the most common events involving the formation of bubbles is dripping, that is the falling of a drop or an object into a quiescent liquid. In the simplest case, when the initial impact sound and the following crown formation are neglected, dripping can be simulated by a single bubble collapsing. However, in some real events (for example, for a large mass falling a volume of liquid) the initial impact sound is indeed perceivable, and the cavity of the bubble is so large that the simple bubble sound model becomes inadequate. This is even more evident in splashing events, when large objects falling into a resting liquid generate many secondary bubbles and droplets due to the mass of liquid displaced by the principal impact event.

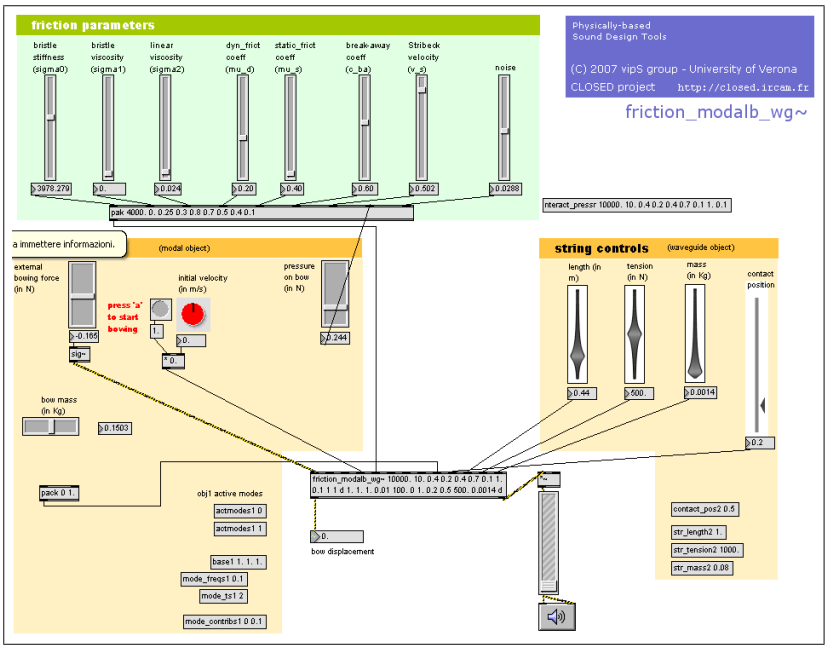


Fig. 2.4. A Max/MSP patch implementing the contact interaction between two solid objects. The object in the center, which is named friction_modalb_wg~ and takes several inputs from sliders or other control devices, is the graphical representation of the corresponding homonym *external* implementing the friction between a modal and a waveguide resonator.

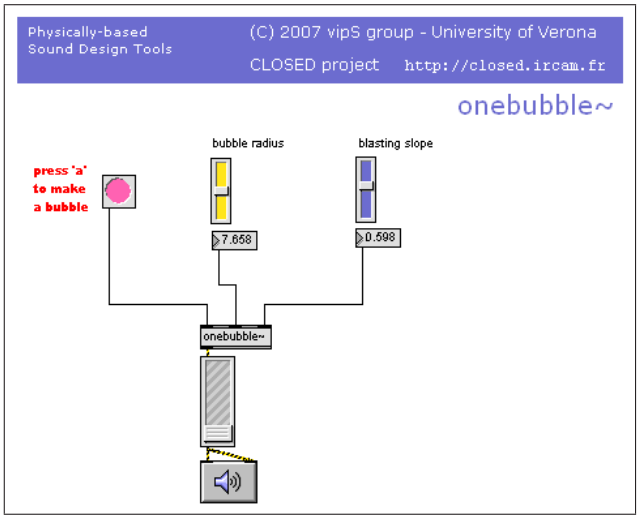


Fig. 2.5. A Max/MSP patch making use of the single bubble model, implemented as the *external* named onebubble~.

The bubble model provides control parameters for the *bubble radius* and the *frequency slope*, which controls the speed for the rise of frequency.

Figure 2.5 shows the help patch for the single bubble model.

Spatialization

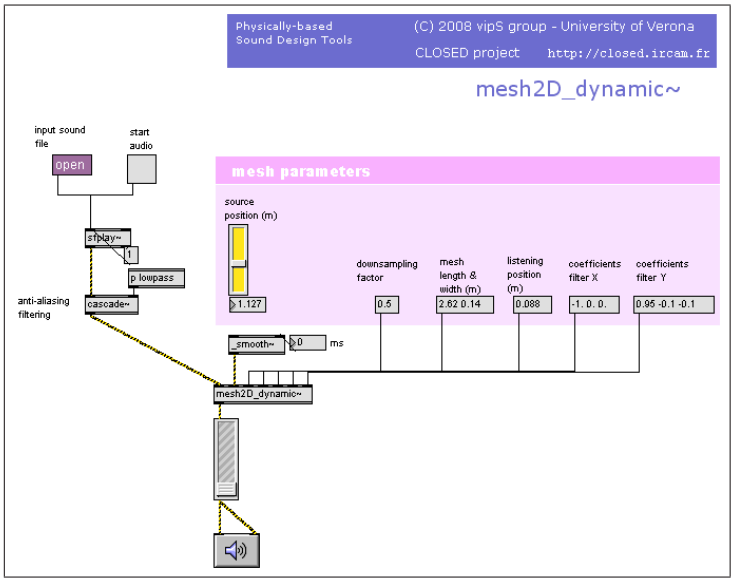


Fig. 2.6. A Max/MSP patch making use of the DWM spatialization model, implemented as the *external* named mesh2D_dynamic~.

In recent years research on the digital waveguide mesh (DWM) has enabled to use discrete-time simulations of acoustic propagation as a model for real-world acoustical spaces [75]. Recent studies [29, 30, 42, 43] have demonstrated the effectiveness of DWM's modeling a rectangular parallelepiped to provide auditory distance cues.

The SDT's sound spatialization model consists of a two-dimensional rectangular DWM [44]. Each internal junction is connected to four other junctions via waveguides, thus providing acoustic wave transmission. Besides, reflections at the mesh boundaries are modeled by digital waveguide filters [101], whose coefficients can be tuned to model specific reflective properties of the surfaces. This structure, modeling a parallelepiped-shaped environment in two dimensions, allows to render dynamic variations of the position of a sound source [45], and consequently provides a tool for interactive manipulation of sound sources along the depth dimension.

Available control parameters are: the *length* and *width* of the parallelepiped, the *source* or alternatively the *listening position* (two versions of the model are provided) along the length of the parallelepiped, the coefficients of two filters

modeling *surface reflections*, and a *downsampling factor* which allows to improve the computational efficiency of the model.

Figure 2.6 shows a help patch for the DWM spatialization model.

2.2.5 High-level models

The attribute "high-level" hereafter generally indicates more complex and structured algorithms, corresponding to somewhat large-scale events, processes or textures. Most SDT's high-level algorithms implement temporal patterns, layering, combinations, or other physically-consistent controls superimposed to low-level models.

Solids

Contacts between solid bodies form a large class of sound-emitting processes in everyday environments. As suggested in Fig. 2.1, many contact interactions may be successfully simulated on the basis of a flexible one-dimensional impact or friction model. Short acoustic events like impacts can strongly gain or change in expressive content when set in an appropriate temporal sequence. Notable examples are provided by the grouping of impacts in bouncing, breaking, crumpling and rolling patterns.

Soft impact

A newly introduced algorithm allows to synthesize the sound of impact on a soft surface, or of soft impact between two surfaces. Indeed, the low-level impact model described in Section 2.2.4 can accurately simulate the impact interaction between a sharp object and another resonator, while it is not able to render, so to say, "softer" impact sounds.

The soft impact algorithm is a rather naive yet effective approach aspiring at filling this gap, while it is not aimed at accurately modeling the physics of contacts between non-pointed objects. Indeed the algorithm is physically-inspired, but it mainly focuses on the actual acoustic result: making use of a dense sequence of tiny signals that excite a resonator, it is able to synthesize soft impacts. In more detail, no actual interaction between objects is simulated, instead the algorithm exploits an *ad hoc* shaped noise burst – representing a force signal – to excite a modal resonator.

Despite being a very simplistic approach, which still has to be validated by experiments, the main idea behind the algorithm can be qualitatively justified considering that non-sharp contacts can be reduced to dense sequences of micro-impacts, thus in a sense discretizing the surfaces of the interacting objects as multiple contact areas. Also, the use of specifically filtered noise signals can be motivated considering that such micro-impacts can have a quasi-random character.

The available control parameters allow to fully control the modal resonator (see Section 2.2.4), plus an ADSR envelope (*attack time*, *decay time*, *sustain gain*, *sustain time*, *release time*) and the *cut frequencies* of two filters (respectively, high- and low-pass) which process the noise burst.

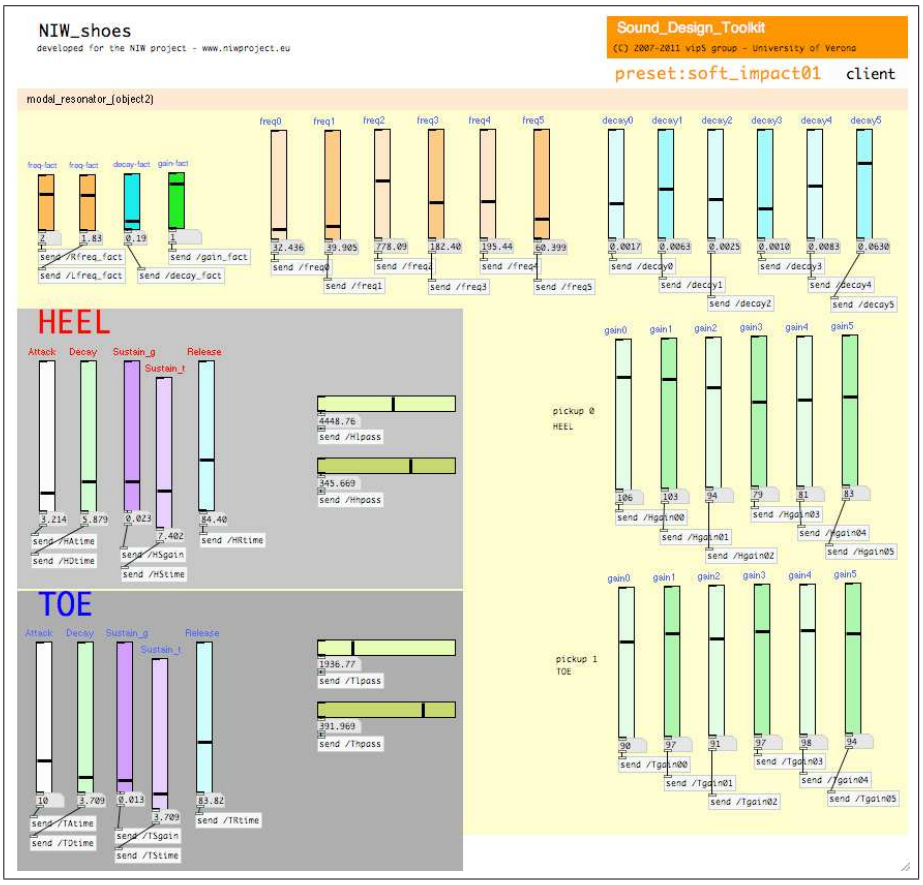


Fig. 2.7. Pd example patch implementing the soft impact model. On gray background are the controls for the envelopes and filters processing the noise burst, while on light orange background are the controls for the modal resonator.

Figure 2.7 shows an example patch developed for the EU-funded project *NIW* (mentioned in the introduction to this thesis), implementing the soft impact model and allowing to simulate the contact between a shoe and the ground, in particular providing two separate envelopes corresponding respectively to the heel and the toe (see also Section 3.3).

Bouncing and Breaking

In [119] it is shown that acoustic stimuli created through the layering of several recordings of collision sounds were identified as bouncing or breaking, depending on the regularity and density of their temporal distribution. These results have been effectively exploited in the SDT making use of the low-level impact model [93].

The bouncing model is basically the result of a constant external force component (like gravity) which is superimposed to the low-level impact model. More specifically, a constant external force is applied to the striker object (represented

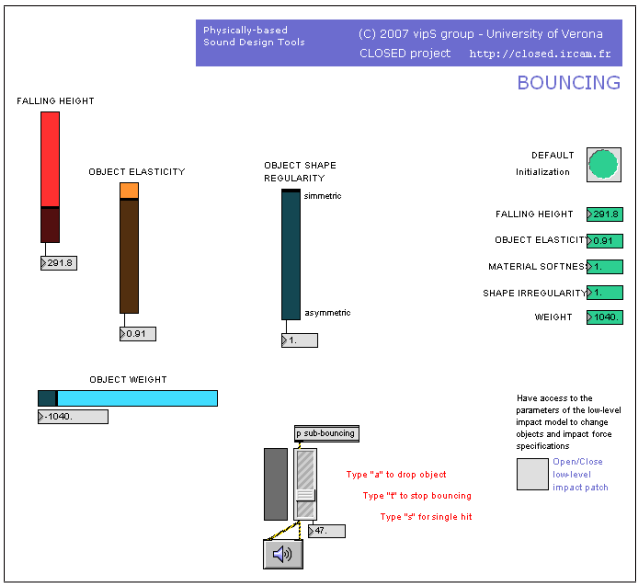


Fig. 2.8. Example patch for Max/MSP implementing the high-level bouncing model.

by the inertial object, that is a point mass), which is directed towards the struck object (modal resonator). The latter simulates a fixed surface which resonates.

The one-dimensionality of the impact model only allows to simulate symmetrical, basically spherical, bouncing objects. A detailed low-level physical simulation of arbitrarily-shaped objects would result in extremely complex controls and heavy computational load. Instead, the simplified high-level modeling of typical bouncing patterns implemented in the SDT leads to an efficient yet ecologically expressive model, still retaining attributes like regularity/irregularity of the bouncing object.

One basic process associated with bouncing is the loss of kinetic energy due to friction and acoustic vibrations. Under the assumption that for a bouncing sphere the loss of energy at each rebound is proportional to the remaining kinetic energy, one obtains a global energy term which decays exponentially with the number of rebounds. As a result, impact velocities and temporal intervals follow exponentially decaying behaviors. The implementation of this basic hypothesis indeed proved to compare favorably with real recordings of bouncing wooden balls.

As for irregular objects, while energy transfers occur between the horizontal, vertical and rotational components, one could assume that only the vertical velocity and displacement effectively affect temporal intervals and velocities of impact. Indeed, the contribution of rotational movements can not be described in a simple way, while the horizontal displacement can be considered as null. On the other hand, energy transfers can be approximately modeled by modulating both temporal intervals and velocities of impact, as deviations from a simple exponentially decaying behavior. Furthermore, the rotation of an irregular object also results in the shifting of its contact points, which can be simulated by modulating the weights of its resonant modes.

For non-spherical shapes with well defined symmetries – for example a cube – the assumptions above can be extended. In these cases transfers of energy between the vertical, horizontal and rotational components can take place in regular patterns, closely related to those of spherical objects.

As for the model implementation, its control parameters are: the initial *falling height*, which corresponds to the time between the first two rebounds; the *weight* of the object; its *elasticity*, which is reflected in an exponential acceleration factor acting on the speed of the temporal sequence; its *symmetry/sphericity*, which is reflected in the range of random deviations of impact velocities.

Figure 2.8 shows an example patch implementing the high-level bouncing model.

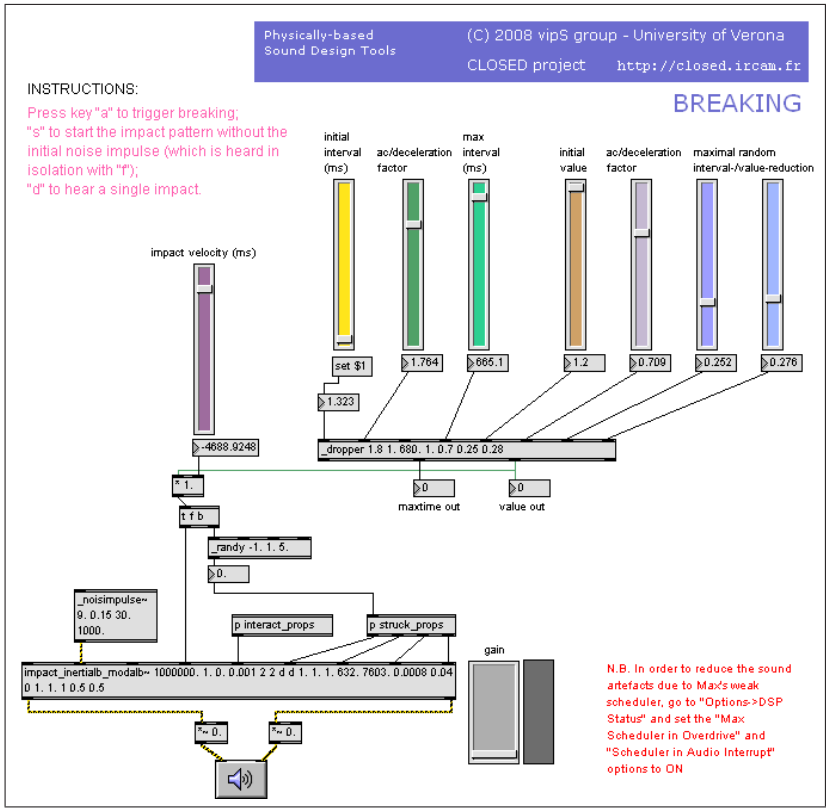


Fig. 2.9. Example patch for Max/MSP implementing the high-level breaking model.

Fragments resulting from rupture usually present highly irregular shapes and are rather anelastic. As a consequence, the bouncing patterns associated with breaking follow a decelerating rather than accelerating progression. Also, fragments

can mutually collide, and the number of such collisions rapidly decreases, starting with a massive initial density.

For the reasons pointed out above, breaking events cannot be described as simple bouncing patterns, however the breaking algorithm makes also use of the high-level control structure used in the bouncing model, with high values of randomness and a quickly decreasing temporal density, that is a behavior opposite to that of bouncing. Following [119], a short noise burst is added at the beginning of the collision pattern in order to emphasize the breaking character of the sound.

It was noted that sequences of impacts with identical temporal structure are less identifiable as breaking events when the resonator is tuned to a metallic character. This may correspond to the fact that in our everyday experience fractures of metal objects are rather rare. Also, extreme mass relations between striker and struck objects lead to more convincing results. Again, this is in accordance with typical situations of breakage: a concrete floor has a practically infinite inertia in comparison to a bottle of glass.

The available controls are: the *impact velocity*, which sets the velocity of the initial (rupturing) impact; the *initial interval*, which sets the time between the first two collision and affects the time between subsequent impacts; two homonym controls set the *deceleration factor* of impact occurrences in two separate randomization processes; the *maximum interval*, which sets the maximum time between impact events; an *initial value*, which sets the seed feeding the randomization process; two controls for the *maximal random interval-reduction* drive two separate randomization processes, namely controlling the maximum random decrement of the time span between impact occurrences.

Figure 2.9 shows an example patch implementing the high-level breaking model.

Rolling

Physical models of rolling are typically complex and computationally expensive. In the SDT a simplified approach has been adopted, which allows real-time implementation of the algorithm while retaining perceptually relevant results and physically-consistent controls.

The heart of the rolling algorithm [91] is the SDT's low-level impact model, which gives access to attributes as the mass or the hardness of the interacting objects. Instead of expanding the features offered by this physical model in order to address the much more complex interactions found in rolling – which would have resulted in higher computational costs, specialization of the model and intricate controls – some higher-level structures were used to reduce the rolling scenario to a sequence of impacts. The impact model implementation chosen for the rolling algorithm makes use of an inertial object (that is a point mass) as the striker, and a modal resonator which simulates a resonating surface. Since the impact model only accounts for one-dimensional interactions (namely, in this case, the normal contact between a rolling ball and a surface), certain macroscopic features, as the global geometry and the transversal velocity, have to be accounted for under perceptual considerations. Also, friction components can be considered comparatively small, and therefore have been discarded.

As anticipated, the rolling model combines different underlying components: the impact model is driven by two pseudo-physical models that simulate respec-

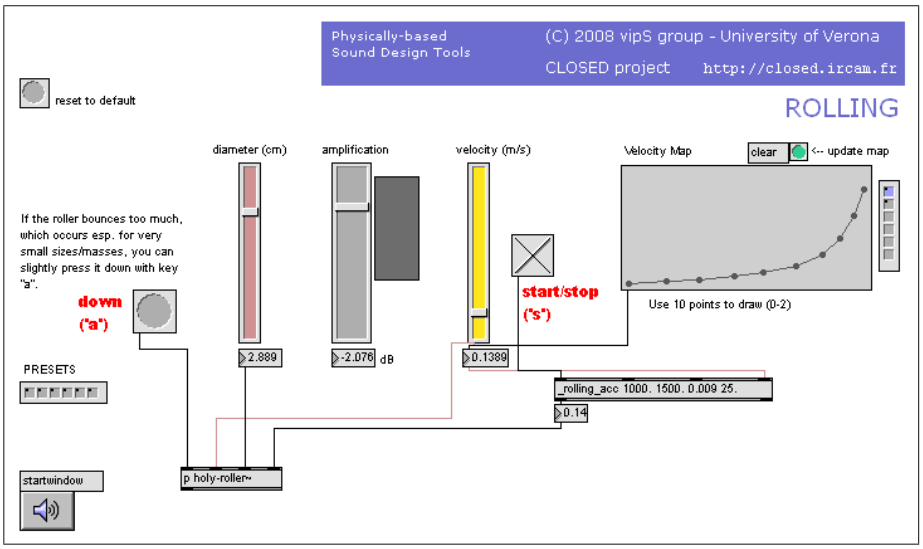


Fig. 2.10. The main Max/MSP example patch for the rolling model. The velocity map represents the acceleration/deceleration curve, which can be drawn by the user. A number of preset curves are provided, and new ones can be saved and recalled.

tively the ball's asymmetries and the local geometry of the contacting surfaces (a ball on a plane surface). These components result in a modulation of, respectively, the external force which is superimposed to the striker object, and the displacement offset between the striker and the struck objects. Both these modulations affect the temporal pattern of micro-impacts.

In more detail, the micro-irregularities of the plane surface are represented by a band-pass filtered noise signal, which was chosen as an computationally efficient and effective solution for simulating such irregularities. The filtered noise signal serves as input for a specifically designed filter which in turn calculates the trajectory of (the center of) the ball. Also, given a surface (that is, a noise signal), the trajectory depends on the ball's diameter.

The rolling algorithm gives access to some ecological attributes of a rolling scenario: the *diameter* and *velocity* of the ball, as well as its *velocity map* can be controlled.

Figure 2.10 shows the main Max/MSP example patch for the rolling model.

Crumpling

Also the crumpling algorithm [41] is the result of an *ad hoc* control layer superimposed to the low-level impact model. Similarly to the soft impact model, the crumpling algorithm does not actually model physical contacts between solid objects but, rather, time sequences of crumpling events, represented by groupings of impact events. These sequences provide data that drive the evolution in time of the parameters of an impact model.

Both the temporal distribution of crumpling events and their own power follow stochastic laws which are derived from physics [60]. Such laws govern 1) the energy

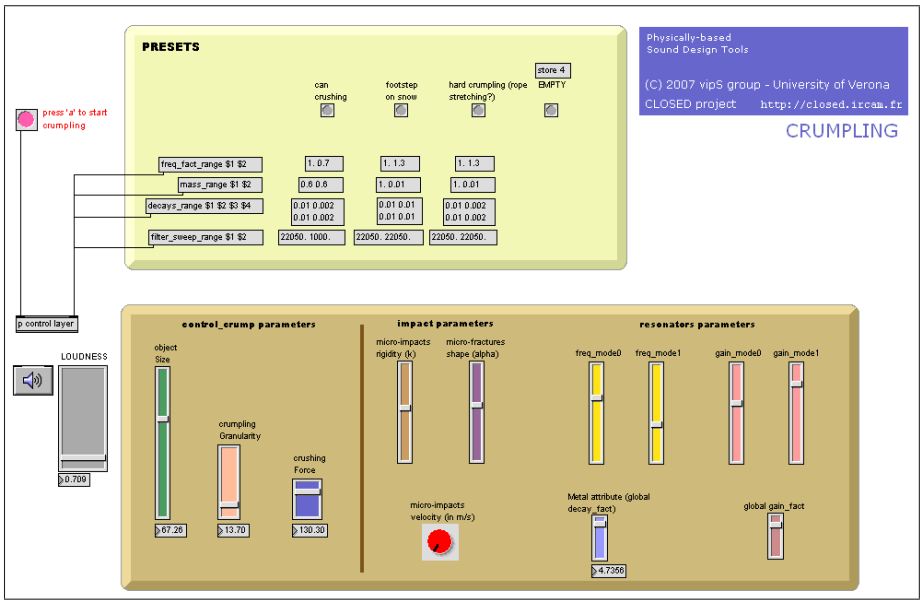


Fig. 2.11. A Max/MSP example patch implementing the revised crumpling model.

dissipation occurring during an impact, and 2) the temporal distribution of adjacent events. Each phenomenon exposes a characteristic parameter, resulting in the control of the average interval between events and the average power of impacts, respectively.

As for the actual implementation of the algorithm, the user is provided with several physically meaningful parameters, which allow to set: the *initial potential energy*, which will be progressively consumed by each impact event proportionally to its own power; the *granularity* of the process, which is reflected in the temporal density of impact events; the *average power* of single impact events.

As a general rule, the same control layer could be exploited by contact models other than the impact model currently being used.

Besides crumpling sounds, by means of this model it is possible to synthesize sounds of crushing (for example, a can being crushed) and footsteps on aggregate surfaces (as ground covered with snow or gravel).

The original crumpling model (as developed for the project *SOb*) has been revised for the SDT in order to extend its features, in particular by including more in-depth controls over the parameters of the interacting objects simulated by the underlying impact model.

Supposing as a simplification that – as the crumpling process advances – micro-collisions occur between fragments of monotonically varying sizes, the *mass* of the striker object, as well as the *decay time* and the *frequency* of the struck object's modes are allowed to vary linearly. The linear trend has been chosen simply as it is of course the easiest way to implement such variation. The user can initialize the model providing it with the variation ranges for the quantities mentioned above,

namely as couples of coefficients which are multiplied for the current value of the target quantity.

As an example, one can assume that the fragments would decrease in size, and consequently both the mass of the striker and the decay times of the resonator's modes would decrease, while the frequencies of the modes increases.

Figure 2.11 shows an example patch for Max/MSP which implements the enhanced crumpling model.

Continuous crumpling

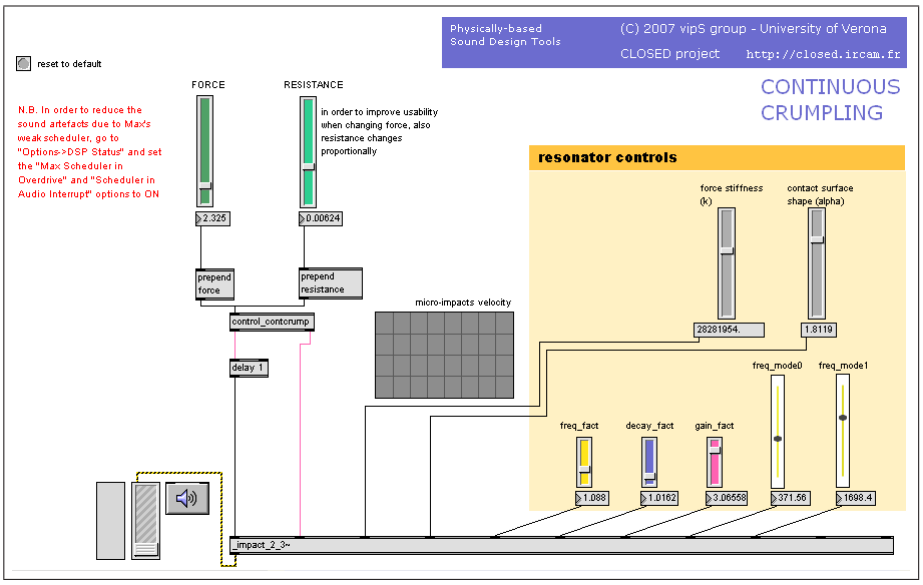


Fig. 2.12. A Max/MSP example patch implementing the continuous-crumpling model.

An alternative version of the model has also been recently added to the library, which allows continuous control over the generation of crumpling events [15].

To a certain extent, the continuous-crumpling model can be seen as a simplified version of the standard crumpling, in that it only provides control over the applied *force*, giving rise to crumpling events and being proportional to their average power, and the *resistance* put up by the material being crumpled, corresponding to the granularity of the latter. Thanks to its external interface, the continuous-crumpling model suits the mapping of continuous gestures to its force parameter: in fact, the model reacts only to variations in the applied force, this way allowing to filter out constant components not reflecting active interactions.

According to the continuous nature of the model, there is not an initial amount of potential energy to be consumed, and therefore the control layer does not automate the variation of parameters such as the mass of the striker, or the frequencies and decay times of the resonator (as seen above). On the other hand, such parameters plus several others are made available to the user via the user interface.

Figure 2.12 shows an example patch for Max/MSP implementing the continuous-crumpling model.

Liquids models

Stream of bubbles

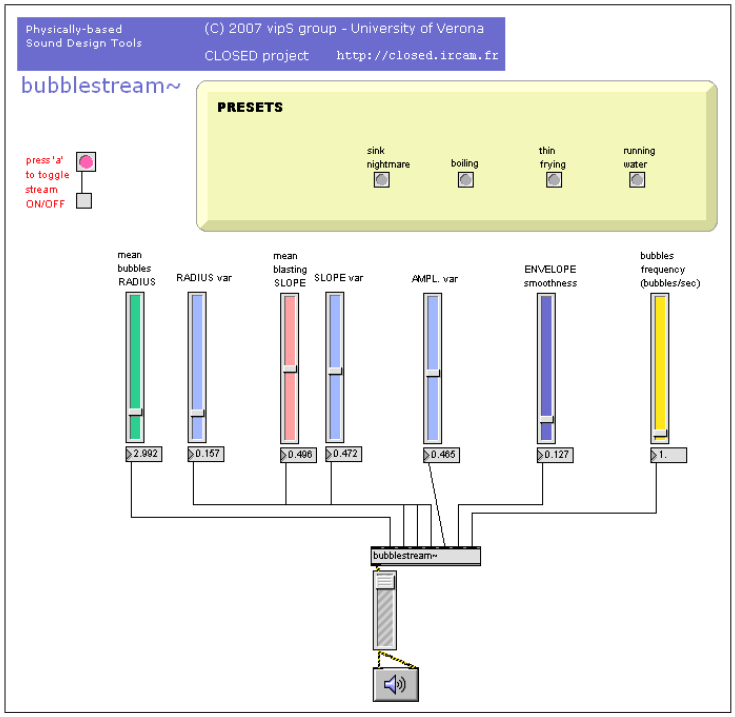


Fig. 2.13. Bubblestream patch

High-level models of complex liquid events have also been considered, which are related to the formation of large quantities of bubbles. Typical phenomena in which distributions of bubbles are observed are, for example: the boiling or frying of liquids, streaming of water, pouring of a liquid into a container or into another quiescent liquid, breaking of waves.

The low-level bubble model, which is described in Section 2.2.4 and simulates a single resonating bubble, serves as the basic element for implementing populations of bubbles. The parameters of the bubbles and their onset instants are set according to given statistical distributions [34]. The so called bubblestream model provides controls over the *frequency* of generation of new bubbles; the *mean* and *variance* of the *radius* and *radius slope*; the bubble *amplitude variance*; the bubble *envelope smoothness*.

Figure 2.13 shows an example patch which offers presets for boiling, frying, and running water sounds.

Splash

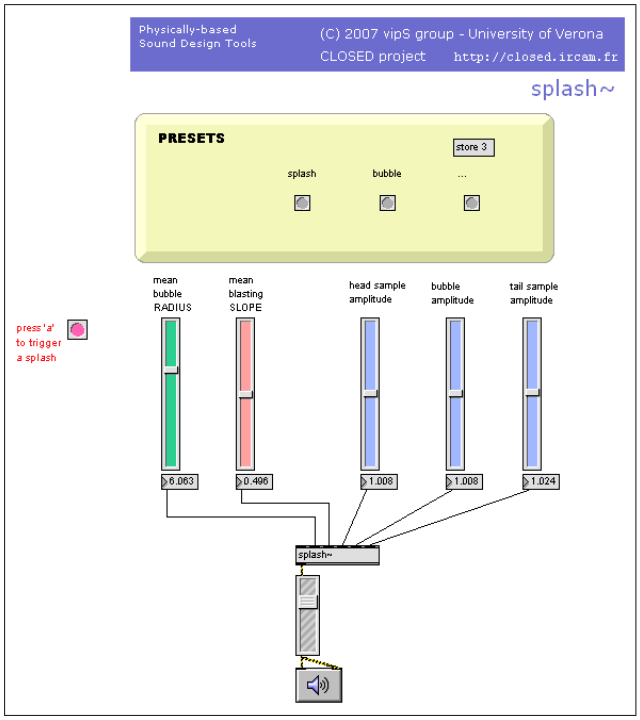


Fig. 2.14. Splash patch

A model has been recently introduced which simulates splash-like events. A splashing sound algorithm has been designed as a temporal sequence of three distinct events: 1) a short initial impact sound, corresponding to the initial interaction of a solid object entering a quiescent liquid; 2) a bubble sound, rendered as detailed in Section 2.2.4; 3) a secondary droplets event texture. In the current implementation, sampled waveforms are used to reproduce the initial and the final sounds, due respectively to the impact and the formation of droplets. In perspective, thanks to the recently introduced soft impact model described above, the initial impact sound component could be simulated through the SDT's synthesis engine.

In addition to the control parameters of the bubble model, the algorithm provides access to the *volume amplitude* of the three components described above.

Figure 2.14 shows an example patch implementing the splash model.

Particles models

A numerical model for fluid simulation known as *smoothed particle hydrodynamics* (SPH) [73] is adopted to synthesize the sound produced by a liquid in motion. According to SPH, fluids are modeled by means of particles representing small volumes of fluid. Each particle is described by a set of physical quantities (namely,

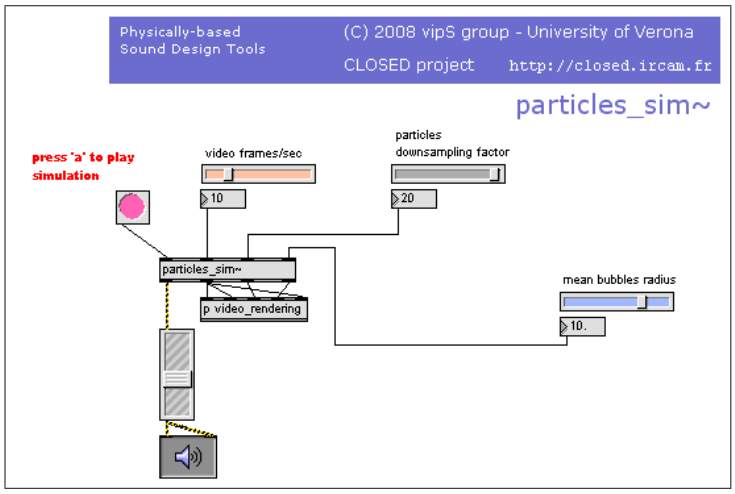


Fig. 2.15. The Max/MSP patch implementing the particle model for fluid simulation.

position, velocity, acceleration, pressure and density) which are updated at each simulation step according to possible interactions with neighbor particles. The interaction is ruled by the Navier-Stokes equations for the conservation of mass and momentum in fluids. Moreover, external forces – such as those due to gravity or collisions with solids objects – can be accounted for in the simulation.

The SDT implementation [35] currently addresses a configuration representing the falling of a liquid volume into an rigid container. Commonly used computation rates for the update of particles (usually 10 to 30 frames per second) are used, and when events are detected for which acoustic emission is predictable, a corresponding acoustic event is triggered. In particular, each sound event is assimilated either to the formation of a single resonating cavity (bubble) under the surface of the liquid (such as in dripping or pouring), or to impacts on the surface (such as those occurring between two liquid volumes, or between a solid and the surface of a liquid).

The mapping from a certain state of the system of particles to audio events was designed considering the available physical parameters describing the particles: for example, the radius of bubbles arising from dripping events is directly related to the mass of the impacting liquid and to the depth of the liquid resting in the container.

Figure 2.15 shows the Max/MSP patch implementing the particle model for fluid simulation.

Other high-level models

Vacuum cleaner

Another recent addition to the SDT's library, the so called vacuum cleaner model, simulates airflow inside a tube having one switchable open/closed termination and one open-ended side. In detail, a pink noise signal is used as input for a

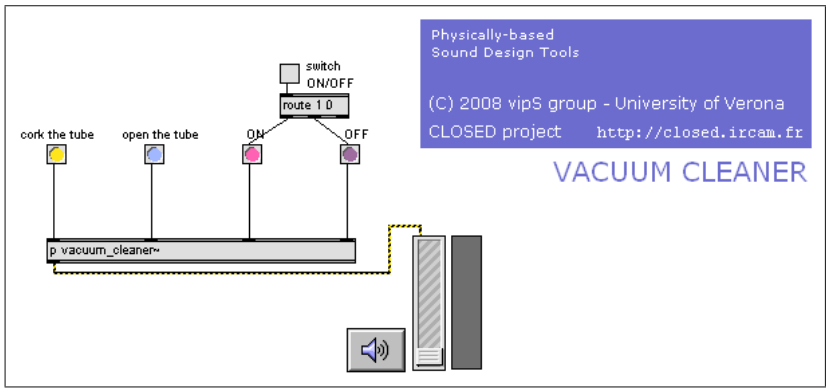


Fig. 2.16. The Max/MSP patch implementing the vacuum cleaner model.

simple model of a tube, this way somewhat simulating the overall effect of motor noise and air getting through the tube. In Fig. 2.17 a detailed diagram of the

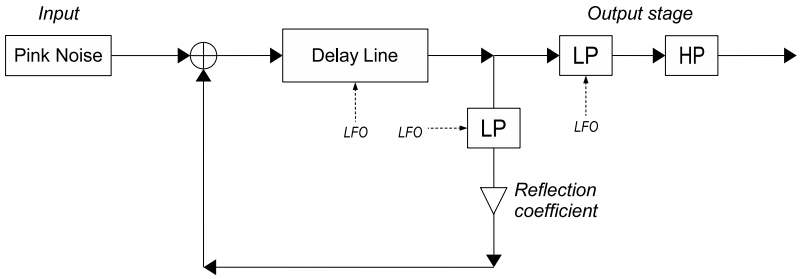


Fig. 2.17. Detailed diagram of the vacuum cleaner model. A delay line, corresponding to the tube length, is in series with a low-pass filter simulating the losses at one termination, and a switchable reflection coefficient accounting for the state (open/closed) of the other end of the tube. A low-pass and a high-pass filter in series further process the audio signal at the output stage. A low frequency oscillator (LFO) allows to slightly vary the length of the delay line and the cutoff frequencies of both the low-pass filters.

vacuum cleaner model is shown, which is based on the digital waveguide [102] paradigm: a delay line, corresponding to the tube length, is in series with a low-pass filter simulating the losses at one termination, and a switchable reflection coefficient accounting for the state (open/closed) of the other end of the tube. In order to refine the sound, at the output stage a low-pass and a high-pass filter in series further process the audio signal. Moreover, a low frequency oscillator (LFO) provides some modulation, namely slightly varying the length of the delay line (that is, the pitch) and the cutoff frequencies of both the low-pass filters.

The example patch shown in Fig. 2.16 implements the model described above within the sub-patch vacuum_cleaner~.

The available controls are: a switch which turns the virtual vacuum cleaner *on and off*; a radio button to *cork the tube*, which triggers the simulation of a corked

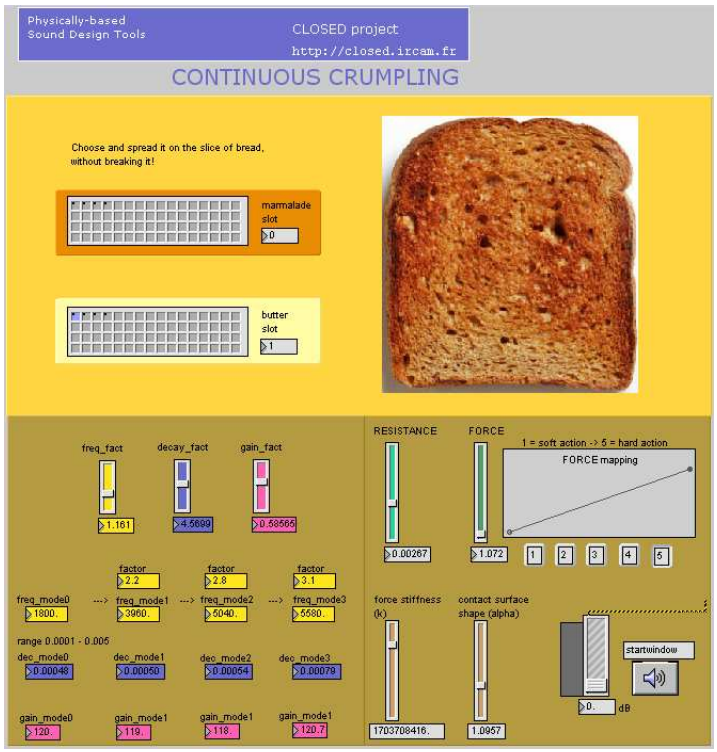


Fig. 2.18. Advanced example patch for the continuous-crumpling model.

tube, namely by setting a negative sign for the reflection coefficient and lowering the cutoff frequencies of the low-pass filters; a radio button to *open the tube*, which triggers the simulation of the opening, namely by setting a positive sign for the reflection coefficient and increasing the cutoff frequencies of the low-pass filters.

2.2.6 Advanced examples and sound palette

Patches for Max/MSP

Exploiting the advanced features offered by Max/MSP, several example patches [25] have been added which show the full potential of the models. For example, such patches may implement complex control layers on top of the models, or may provide an intuitive graphical interface which can be accessed with either the computer keyboard or the mouse. Figure 2.18 shows an advanced example patch for the continuous-crumpling model (see Section 2.2.5).

Also, in order to give a hint of the extremely wide and expressive sound palette which is achievable by the SDT, several presets [25] – corresponding to notable configurations of parameters – have been provided for the models. Figure 2.19 shows the Max/MSP patch illustrating the sound palette for the bubblestream model (see Section 2.2.5).

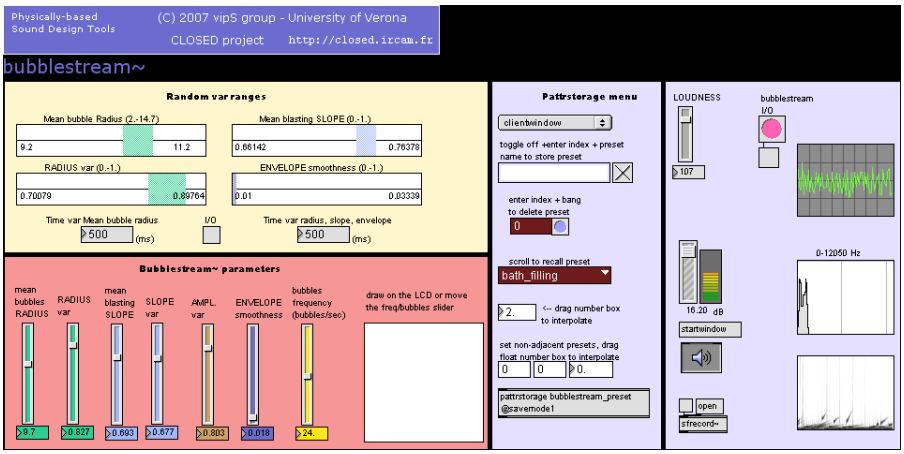


Fig. 2.19. Sound palette patch for the bubblestream model.

In order to further expand the interaction possibilities, a dynamic morphing system has been implemented which interpolates the values of parameters while changing presets, thus allowing both smooth transitions and intermediate explorations of sound presets. In this way one can associate the user's gesture to transformations from a configuration of parameters to another one, thus for example changing the simulated material, or the interaction properties. In this regard, it is worth noticing that high perceptual effectiveness was found when leaving the parameters of the resonators (*structural invariants* [119]) unaltered, while interpolating the interaction properties (*transformational invariants* [119]). Instead, associating gestures to changes in the parameters of the resonators generally resulted in unnatural and artificial sounds. Of course, such results are in agreement with the everyday experience, where only rarely a sounding object changes its nature during the interaction (for example in breaking events).

Presets can be saved as XML files, thus allowing them to be read, modified or generated also from other XML-compatible software.

Patches for Pd

Recently, several example patches for Pd have been added, which present some of the models in a client-server fashion (see for example Fig. 2.7). Two are the main motivations which justify such approach:

- Pd does not offer intuitive and efficient data structures for managing presets, and therefore it was decided to exploit its "wireless" functionalities to enable the connection of one server patch – basically, hosting the sound model – to several client patches – hosting a user interface which also initializes the model's parameters as soon as it is loaded – each representing a different sound preset. In other words, the user loads the server patch and then choses a client patch in order to change preset.
- Some applications require that the sound models reside on a machine, while the user interface stands on another one. This is especially useful when making

use of ultra-mobile or embedded systems (for example, a mobile phone of the latest generation) that would send control messages to a server patch, hosting a sound model, via wireless connection. To this end, a set of examples based on OSC[14] have been included in the SDT.[15]

2.2.7 User interface

Help patches

As mentioned in Section 2.2.1, each low-level model and part of the high-level ones are implemented as *externals* for Pd and Max/MSP, each corresponding to an object for their graphical programming environments. The best way to understand how the supplied *externals* work is to have a look at their respective help patches.

Due to the modular structure of the SDT, several implementations of solid contact models are possible (some of which are already provided in the package), which involve different objects and interactors (see Section 2.2.4): as an example, Fig. 2.3 and 2.4 show the help patches respectively for the impact model implementation where two modal objects collide, and for the friction model implementation where a modal object rubs a waveguide resonator. All the help patches corresponding to solid contact models share a common design following Fig. 2.2: the panels on the left and right sides (having orange background) show control parameters respectively for object1 and object2, while those in the center (having green background) are for the interactor. Generally, in such help patches, the controls made available for object1 only operate on a subset of its own properties (basically for screen space's sake), yet they allow to trigger the interaction with object2. Conversely, the controls provided for both the interactor and object2 operate on the whole set of their available physical-geometric properties.

A similar design is retained – whenever possible – throughout the library of help patches offered by the SDT.

Front-end

Since the SDT's sound synthesis algorithms generally have strong physic foundations and, as a consequence, their control parameters have a physical counterpart, one may assume that they should in principle be easy to interpret. However such physical relations can be quite obscure to the non specialists, and moreover the algorithms – representing only approximations of real acoustic phenomena – may not reflect the behavior of real systems for the whole parameter space.

Such issues have been partially tackled with the latest addition to the package, aiming at making the SDT an accessible and flexible tool for sound designers, without requiring an in-depth knowledge of the Max/MSP or Pd environments, and facilitating its use in real-time interactive applications. A high-level interface have been developed [25] which offers many facilities in terms of connectivity, mapping and usability of the models.

[14] See footnote 9.

[15] It is shown in Section 2.2.7 that the advanced front-end provided for Max/MSP also implements OSC communication.

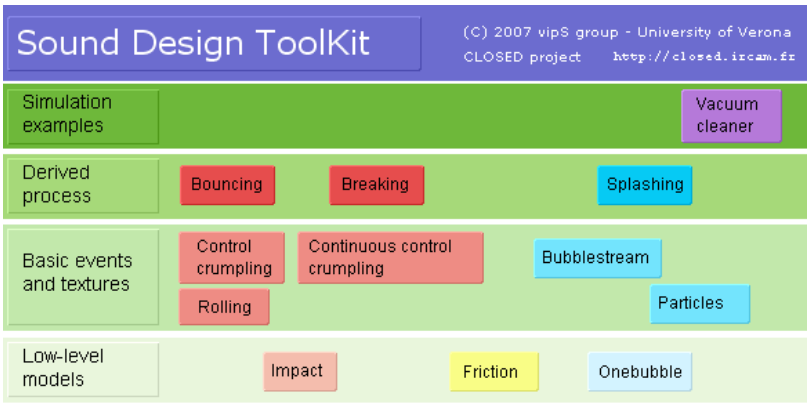


Fig. 2.20. The main SDT's front-end showing the taxonomic organization of sound synthesis models.

Based on the taxonomic organization of everyday sound pictured in Fig. 2.1, a unified interface is provided as a main patch (for Max/MSP only) giving access to the entire sound palette offered by the SDT:[16] when the patch `SDT.mxb`, shown in Fig. 2.20, is started, a front-end shows-up displaying the available sound models among low-level events and sound textures, while derived processes can be obtained through their combination.

Usually, when first approaching a software application, users prefer to *use* it, instead of spending their time trying to understand how it works or why it does not. Moreover, given the minimum basic knowledge required to start, users typically demand for clear and simple interfaces which provide effective information about control parameters and the current state of the system.

Starting with the impact and friction models, a first attempt has been made to uniform the GUI size, the position of the control parameters, and the use of colors, thus allowing to better distinguish the functionalities of each module. Secondly, since the meaning of physical parameters' values can be sometimes confusing and hard to understand, sliders and dials have been scaled in to a more straightforward and readable 0-100 range. Finally, the naming of parameters has been partly revised with the aim of stressing the corresponding perceptual effect: for example, the original parameter of "stiffness" has been renamed as "material elasticity", which gives a better understanding of its incidence on the characteristics of the material.

Since physically-based sound synthesis generally requires the user to tune a large set of variables, the new GUI is intended to provide him/her with a functional hierarchy of parameters, thus facilitating their intelligibility while retaining the access to low-level controls. The interface for each model is partitioned into a low-level and high-level one:

- High-level interfaces (see Fig. 2.21) are intended to give access to the most relevant control parameters, in such a way that they can be directly manipulated

[16] The development of the main interface is still in progress: at the time of writing only a subset of the models are accessible from it.

with external devices (for examples, through sensors or hardware controllers). To this end, high-level interfaces are split into three main sections: 1) physical parameters; 2) temporal control; 3) timbral palette. The *physical parameters* section encompasses the most direct and effective parameters: for instance, for an impact model the material elasticity and the shape of the contacting surfaces (as for the interaction), the hammer mass (as for object1), and global factors affecting frequency, decay time and gain (as for object2).
Depending on the sound model, the *temporal control* section offers sequencing tools that allow to simulate gestures or patterns, resulting especially useful during the sound design process. As an example, for the friction model this section allows control over the velocity and the rhythm of rubbing, and the duration of the exerted pressure.
Finally, the *timbral palette* section allows to manage, store, save, recall, delete or interpolate sound presets.

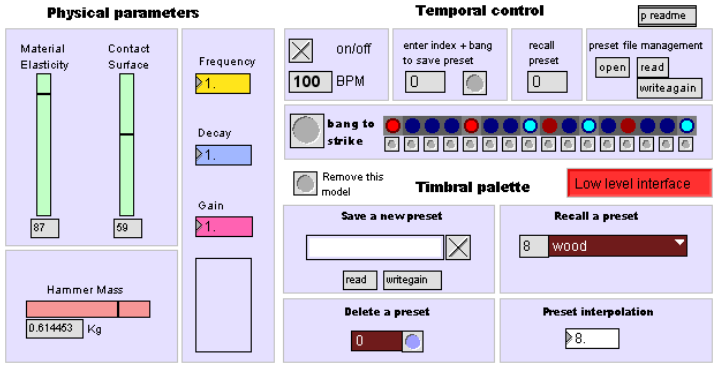


Fig. 2.21. High-level interface for the impact model: physical parameters, temporal control and timbral palette sections.

- The low-level interface (see Fig. 2.22) gives access to the whole set of controls of the models, for further tuning and refining the sound. Among the others, it is possible to activate/deactivate or mute the modes of resonance and modify their individual frequency, decay time and gain, thus allowing to create more refined and complex sounds.

The front-end is also provided with a set of dedicated side-tools that allow to easily manage projects, and that are of help when working with sensing devices.

Recallable patches are configured as *alias* where all variables and connections are supplied as changeable arguments. As a result, it is possible to load multiple independent instances of the same sound model within the same session. Also, it is possible to remove or replace sound models at user's pleasure.

Each sound model added from the front-end is provided with some additional features, as independent MIDI and OSC[17] interfaces, volume control and audio recording tools. These are automatically generated and grouped in a separated

[17] See footnotes 8 and 9, respectively.

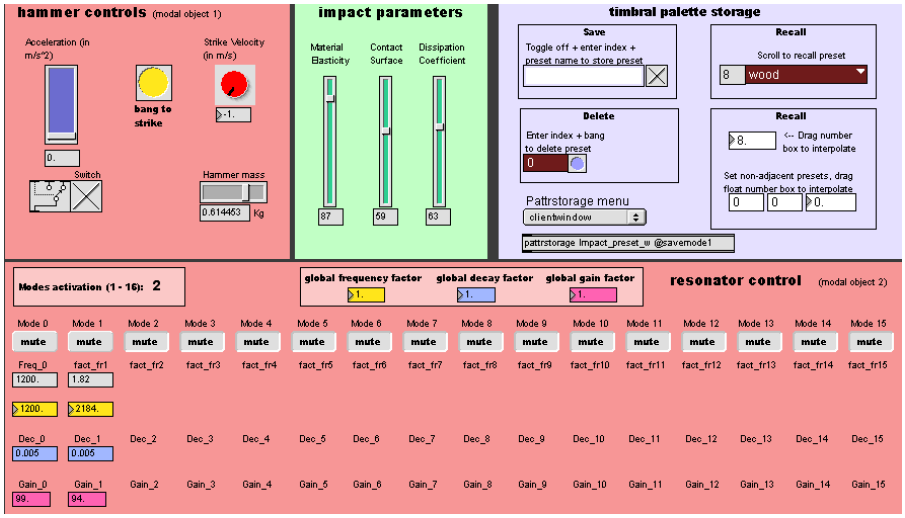


Fig. 2.22. The low-level interface for the impact model, gives access to all the available parameters.

window providing a mixer with separate volume and level indicator for each sound model instance.

Starting from this framework, the aim is to progressively make the SDT as "plug & play" as possible, in order to effectively exploit the interactive characteristics of the physically-based sound synthesis paradigm.

3

Applications in interactive sonification

In order to move from the desktop to the actual realm of physical computing, a variety of sensors have been used to control some of the models offered by the SDT. The possibility to map the control parameters of the models to continuous physical interactions, and their correspondence to physical and geometric properties, make the SDT a powerful tool for designing expressive sound feedback in real-time applications.

For instance, within the project *CLOSED* several scenarios and everyday objects were provided with sensors and sonically augmented,[1] and such in-the-field investigation on the role of continuous sound feedback in the interaction loop [28,67] allowed to test the models and start developing paradigms and a vocabulary for the use of the SDT.

Examples of sonically augmented artifacts can be found already in the consumer market and in everyday life.[2] Just to name a few: the mother of them all is probably the Geiger counter, which produces static-like sounds when radioactivity is detected; in Bialetti's Moka Sound, a melody alerts the user when the coffee is ready and the stove should be turned off; Apple's iPod and Mighty Mouse substitute mechanical noises with synthetic feedback, respectively for the virtual rotary dial and the scroller ball; parking sensors, present in many recent vehicles, are usually associated to synthetic beeps whose frequency increases as approaching an obstacle; similarly, in many cities pedestrian traffic lights are provided with auditory feedback alerting pedestrians if it is safe to cross the street or not.

As the examples above point out, sound can be used for different purposes, with different degrees of functionality. For example: adding auditory feedback where it was previously absent or where other sensory information is insufficient/inadequate/uncomfortable (Geiger counter, parking sensors and, especially for blind people, traffic lights) can help improving the user performance; substituting the sound usually coming from mechanical devices (iPod, Mighty Mouse), thus allowing the maker to personalize the sound and save on mechanical parts;

[1] For further details please refer to the public deliverables 3.1, 3.2, 3.3 available from `http://closed.ircam.fr/deliverables.html`

[2] A growing list with several examples of sonically augmented artifacts is available at `http://trac.sme-ccppd.org/SID/wiki/WG2Product/Examples`, and is also open to external contributions.

adding sound to a process which is already sonically relevant (Moka Sound), in order to improve the user performance, while also making a product fancier.

In this chapter, some prototype interactive applications are described, which have been recently developed exploiting the tools offered by the SDT towards sonic interaction design and interactive sonification.

3.1 The Gamelunch

The Gamelunch[3] [26, 27, 87] is a sonically augmented dining table, which was originally developed within the EU-funded project *CLOSED* (which is reviewed in the introduction to this thesis). The main aim of the Gamelunch is to let users experience the importance of interaction-coherent sounds in everyday-life acts, as that of sitting at a table and having lunch. Indeed, with its set of immediate and natural gestures and actions, the dining scenario sets a fertile context for the investigation of interactive sound feedback. In this respect, the Gamelunch exploits the power and flexibility of physically-based sound models towards the investigation of the closed loop between interaction, sound and emotion.

3.1.1 Study of the dining scenario

As described in Chapter 2, most of the sound synthesis algorithms provided by the SDT are based on modeling sound sources in terms of their physical behavior, thus potentially providing a natural mapping between gestures and the control parameters of sound models. In this perspective, the sound design process has to undergo a precise decomposition of events occurring in the action, and to this end continuous (inter)actions – such as cutting, piercing, sticking, drinking, pouring, grasping, stirring, mixing – have been analyzed focusing on their sound production mechanisms and, of course, resulting sounds. Complementarily to such "source behavior" approach to sounding objects [96], the investigation was also carried out in terms of interaction *gestalts* [68] and complexity of the corresponding sensory-motor skills. Following a basic design approach [4], various sensors-augmented kitchenware have been sketched around well defined themes and constraints, exploring the interplay between sensory channels, and finally starting to develop a logic and a vocabulary tailored for the SDT.[4]

A number of qualitative and conceptual questions have arisen with regard to mapping strategies for coupling 1) continuous gestures, 2) sensed data and 3) control of sound synthesis parameters.

In the final prototype, pictured in Fig. 3.1, continuous interaction gestures are captured by means of different sensors embedded in both the table and kitchenware, providing data that are coherently mapped onto several physically-based sound synthesis algorithms from the SDT. Sound feedback is provided by means of loudspeakers which are inserted in custom-made holes under the table.

[3] Pronounced *'gamə*-lunch, similarly to the Balinese musical ensemble *gamelan* which inspired the first prototype of the Gamelunch.

[4] The experiences and activities carried out around the sonically augmented dining scenario have been collected and published on the website: `www.soundobject.org/BasicSID`.

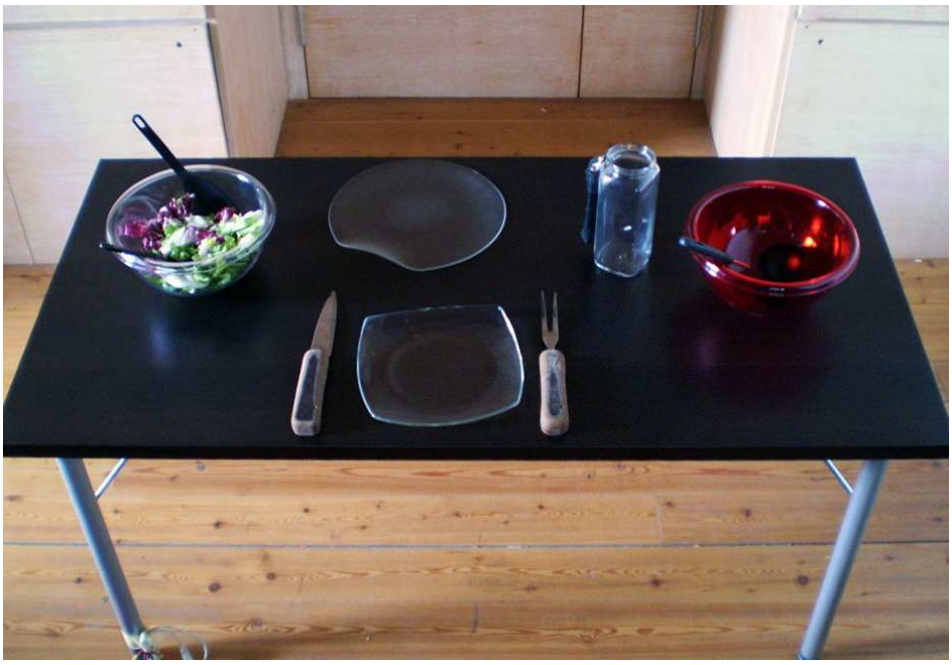


Fig. 3.1. The final prototype of the Gamelunch. Various sensors are embedded in both the table and kitchenware. Loudspeakers for sound feedback are embedded in custom-made holes under the table.

3.1.2 Enaction and sound feedback

Considering sound as the acoustical result of interactions with and between objects, it has been assumed that a direct link is present between sound feedback and causality, and therefore the use of artificial sound feedbacks can either emphasize or contradict causality. Based on this remark, the Gamelunch offers a way to investigate the loop "interaction-sound-emotion", in a way exploiting the principle of contradiction: while performing usual dining actions – such as cutting and slicing, dressing the salad, pouring beverages – the user encounters contradicting and unexpected sound feedbacks, thus experiencing *per absurdum* the importance of interaction-consistent sound in everyday-life acts. In more detail, in order to influence the coordination of different sensory channels in terms of, for example, the perceived effort or object/material attributes, several misleading sonic counter-actions have been realized by bending functional, aesthetic and emotional features of usual natural sound feedbacks. It was found that the distortion of the auditory display channel results in *enactively*[5] affecting emotional responses and expectations in the "perception-action" loop. On the other hand, by ensuring energy consistency between gestures and the resulting sound, the feeling of causal, continuous control of the action is maintained.

[5] *Enactive* knowledge depends on the action-perception loop: action reveals information, which guides further action, which reveals additional information, and so on. See `http://www.enactivenetwork.org/` for further details.

Two kinds of contradiction have been taken into account: one concerning the temporal development of actions (that is, related to *transformational invariants*), and the other one deeply connected with the nature of materials being manipulated (that is, related to *structural invariants* [119]). An example of manipulation of transformational invariants is provided by the sonification of the jug, where the act of pouring water is considered: a friction sound is used to sonify the jug tilt, thus giving the feeling of a resisting force that opposes the natural effects of the liquid pouring and the lightening of the jug. Another example showing the contradictions arising from the manipulation of structural invariants is provided by the action of stirring soup in a bowl: in this case, the sound feedback is similar to that of a tool mixing tinware or sand, oppositely to the fact that nothing solid is present in the bowl. Likewise, the act of piercing with a fork is associated with a squeaking/rubbing sound feedback which alters the perceived consistency and quality of the food [123].

The approach to embodied[6] sound interaction [28] makes the Gamelunch different from other current table-based interactive sonic devices. Most of existing tabletop-like tangible interfaces [22], such as the *reacTable* [63] and other related devices,[7] act as controllers for mainly musical purposes, thus focusing on and questioning about expression issues. Another example, the *Table Recorder*,[8] is a sonically augmented table whose concept and realization cleverly couples interaction and real everyday sounding objects such as glasses, cans, dishes, etc.

3.1.3 Realization

The Gamelunch was conceived as a self-explanatory experience (that is not requiring any apprenticeship by the user) of the role played by sounds in gestures related to the dining scenario. Hereafter is an overview of the sonification system implemented as a set of sonically augmented/altered objects, by linking the data coming from sensors – embedded both in the table and kitchenware – with the control parameters of several physically-based sound synthesis algorithms provided by the SDT. The sound design process started with focusing on possible interactions and expected sound feedback, ending up with the manipulation of physically-meaningful parameters of the SDT's models in order to produce and highlight perceptual-cognitive contradictions.

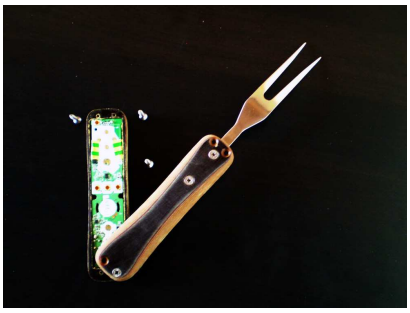
The fork: (see Fig. 3.2(a)) a friction/squeaking sound, similar to that coming from a wooden board bending, sonifies the action of sticking the fork in food. The sensation of weight of the fork and consistency of the food being pierced is warped as if the food, or the dish and table below it, were springy.

The knife: (see Fig. 3.2(b)) as the arm moves forward and backward and the wrist twists, the action of cutting is sonified as the sound of friction on a wrinkled plastic surface. The resulting sensation is that of a hard and fatiguing action, affecting and distorting the feeling of the consistency of the material

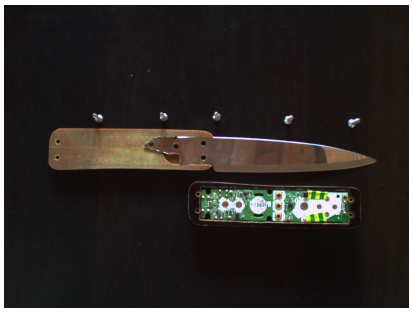
[6] *Embodiment* is a concept related to *enaction* (see footnote 5), and is the result of a learning process deriving from perceiving by doing and doing by perceiving.

[7] `http://mtg.upf.edu/reactable/?related`

[8] `http://www.fregment.com/?cat=0&p=7868`



(a) The fork, opened up in order to show the embedded board which is taken from a Wii Remote controller.



(b) The knife, opened up in order to show the embedded board which is taken from a Wii Remote controller.

(c) Fork and knife, laid on a glass dish. A force sensor is embedded into the table below the dish, sensing the pressure exerted on it.

Fig. 3.2. The Gamelunch's cutlery.

being cut. Using, as the "food" to be pierced and cut by the fork and knife, soft modeling clay molded into cubes, cylinders, and other regular shapes, the material response is in contradiction with the squeaky sonification associated to those actions.

The salad bowl: (see Fig. 3.3(a)) the action of tossing some salad is sonified by cartoon-like sounds of a liquid dripping and boiling. The resulting sensation is that of turning something insubstantial, which either drips or gurgles.

The jug: (see Fig. 3.3(b)) the corresponding action is that of pouring liquids and consists in tilting the jug by moving both the wrist and the arm in order to control the flow of liquid. A continuous friction/braking sound feedback alters the natural perception by generating the sensation of an opposing force which resists the action of pouring. As a result, the smooth flowing of the liquid, and the subsequent lightening of the jug, are in way contradicted by the sound feedback.

(a) The salad bowl stands on a light sensor embedded in the table, which sends data according to the variations of lighting due to the movement of the salad.

(b) The jug. A board from a Wii Remote is embedded into the handle and covered with rubber.

(c) Oil and vinegar cruets, each holding a Wii Remote board inserted in a waterproof rubber case.

(d) The tureen lays on a magnetic field sensor, while two thin magnet stripes are glued to the bottom of the ladle, thus providing data as the ladle moves inside the tureen.

Fig. 3.3. The Gamelunch's kitchenware.

The oil and vinegar cruets: (see Fig. 3.3(c)) the actions of shaking the cruets for spilling oil and vinegar are sonified by means of granular sounds. In detail, the oil cruet is sonified as if a grainy sand-like substance was flowing from it, whereas the spilling of vinegar is associated to the sound of pebbles bumping inside the bottle. Again, these sonification examples allow to alter the perception of the materials/substances being manipulated.

The tureen: (see Fig. 3.3(d)) the action of stirring is considered. The rotation of the ladle inside the liquid content of the tureen is sonified as the sound of solid objects (tin cans, hollow wooden sticks) bumping against each other, thus altering the perception of stirring a smooth liquid.

The cutlery, the jug and the oil and vinegar cruets each make use of a Nintendo Wii Remote[9] controller, which provides an effective way to wirelessly interface its 3D accelerometer to the control parameters of the SDT's models. In detail, the circuit boards of several Wii Remote's have been removed from their plastic chassis, so that the fork, knife and the jug can embed the boards in their handles, while each cruet holds a waterproof rubber case containing a board. The Wii Remote's boards are connected via Bluetooth to a laptop running *ad hoc* developed Max/MSP patches based on the SDT.

The remaining kitchenware is augmented with wired sensors connected to a couple of acquiring boards obtained by hacking two analog game-pads: in this way the analog signals coming from the sensors are converted to 8-bit digital data which are mapped onto different control parameters of the SDT's models. The transparent salad bowl lays on a light sensor embedded in the table, which sends data according to the variations of lighting due to the movement of the salad. The tureen lays on a magnetic field sensor, while two thin magnet stripes are glued to the bottom of the ladle, thus providing data as the ladle moves inside the tureen. Two force sensors are embedded into the table in correspondence of two dishes, sensing the pressure exerted on the dishes while using the fork and knife: pressure data are used together with data coming from the accelerometers in the cutlery to control some friction sound models.

In order to improve the immersiveness[10] and embodiment characteristics of the Gamelunch, the sound is diffused by means of five loudspeakers embedded in the table (see Fig. 3.4), each corresponding to a separate audio channel. In this way sound feedback can be spatialized and distributed throughout the area covered by table.

3.1.4 Results

The Gamelunch represents a self-contained prototype of interactive sonification system which addresses a specific scenario. Since its first realization in early 2007, the Gamelunch has evolved as it was presented and demoed in workshops[11,12] [46], exhibitions [26] and conferences [87] where people were able to directly experience the system.

Even though the investigation is currently limited to a qualitative approach, encouragingly users have so far agreed on the effectiveness of the sonic distortions provided as they allowed them to become aware of the importance of sound feedback in everyday actions. As a side note, it has been noticed as in some occasions the contradictory sound feedback stimulates new ways of manipulating the artifacts: some users overrode the functional space and started challenging the interaction with artifacts in an expressive manner, thus creatively exploring and pushing the boundaries of the designed sonic space.

[9] The handed wireless controller for Nintendo's Wii console. Thanks to Masayuki Akamatsu for his [aka.wiiremote] Max object (available at `http://www.iamas.ac.jp/~aka/max/`).

[10] See Chapter 2, note 1.

[11] `http://sonic.wikispaces.com/`

[12] `http://www.newbasicdesign.it/`

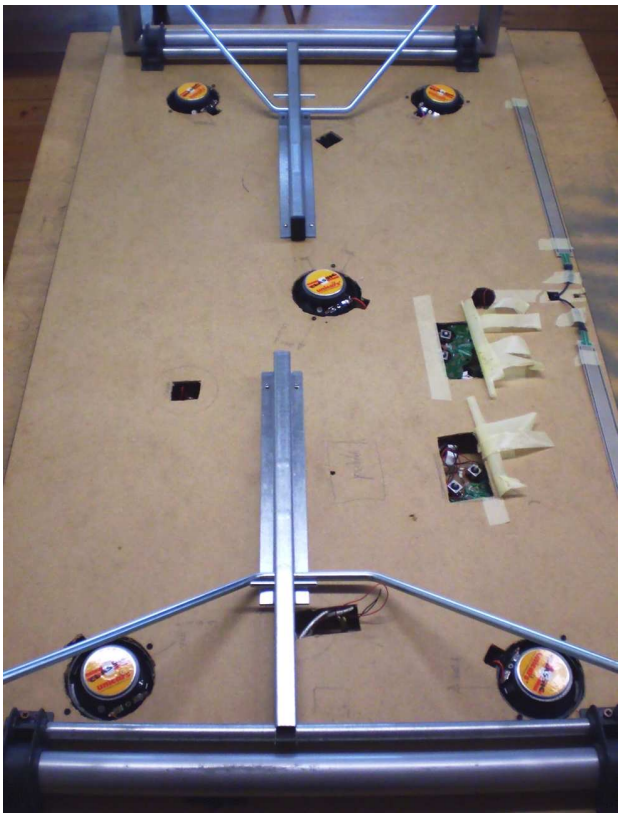


Fig. 3.4. The underside of the Gamelunch, showing the holes hosting the five loudspeakers and the two game-pad's circuit boards.

3.2 DepThrow

DepThrow [82,83] is an interactive audio-only game, designed around the auditory perception of distance and the use of physically-based models for the simulation of the dynamics, sound source, and acoustical environment.

The game consists in throwing a virtual sounding object inside a virtual open-ended tube which is inclined. The task is to throw the object as far as possible without letting it fall out at the far end of the tube, that is the user need to adjust the initial velocity applied to the object. Information about the position of the object inside the tube is provided as continuous audio feedback. The user performance is closely related to her/his ability to perceive the dynamic distance of the object in the virtual tube, therefore this game represents a potential tool for exploring the usability of auditory distance information in interaction design.

3.2.1 Architecture of the interface

The game prototype has been developed as a complex Max/MSP patch where three interacting physically-based models from the SDT enable to simulate, respectively,

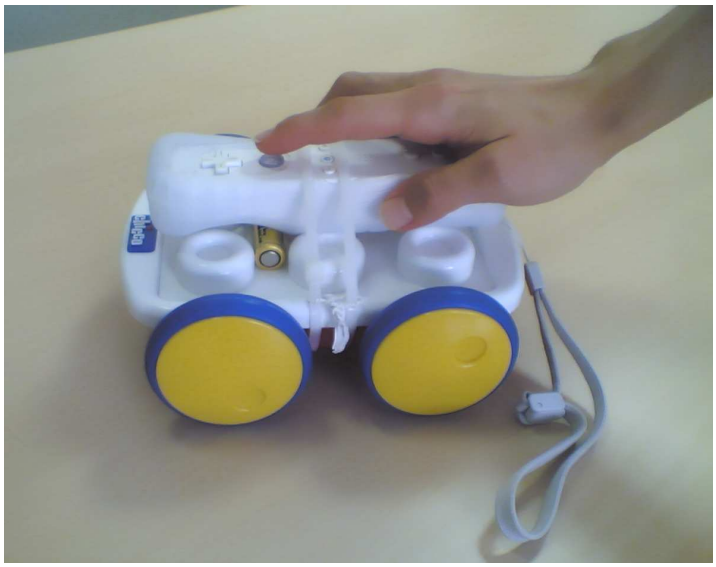


Fig. 3.5. The toy trolley mounting a Wii Remote controller, which is used as input interface for the game.

the dynamics of the environment (an inclined plane), the sound source (a small trolley, represented by a rolling sound), and to render the acoustical properties of the environment (an open-ended tube).

The action of throwing is performed by drawing a toy trolley, on top of which a Wii Remote controller is mounted (see Fig. 3.5). The Wii Remote provides – among other things – a 3D accelerometer, a vibration actuator, several buttons, and Bluetooth connectivity.

The three physically-based models are connected in a top-down chain structure. Figure 3.6 shows the main Max/MSP patch implementing the application. The force applied to the virtual trolley is captured by means of the Wii Remote's accelerometer, providing the input to the computation (via integration) of the initial velocity with which the trolley is getting thrown inside the inclined tube. Then, the initial velocity provides the input to the simulation of the inclined plane where the force of gravity is taken into account. The resulting velocity of the trolley moving along the inclined plane drives the real-time synthesis of a rolling sound, which constitutes the audio input to the spatialization process simulating the open-ended tube. The distance currently covered, which is computed by the inclined plane model, controls the distance parameter of the spatialization model, thus conveying auditory information about the relative position of the sound source inside the virtual tube.

The vibration actuator of the Wii Remote is exploited for providing tactile feedback: namely, when the object comes back to the user's hand, a short vibration burst is given.

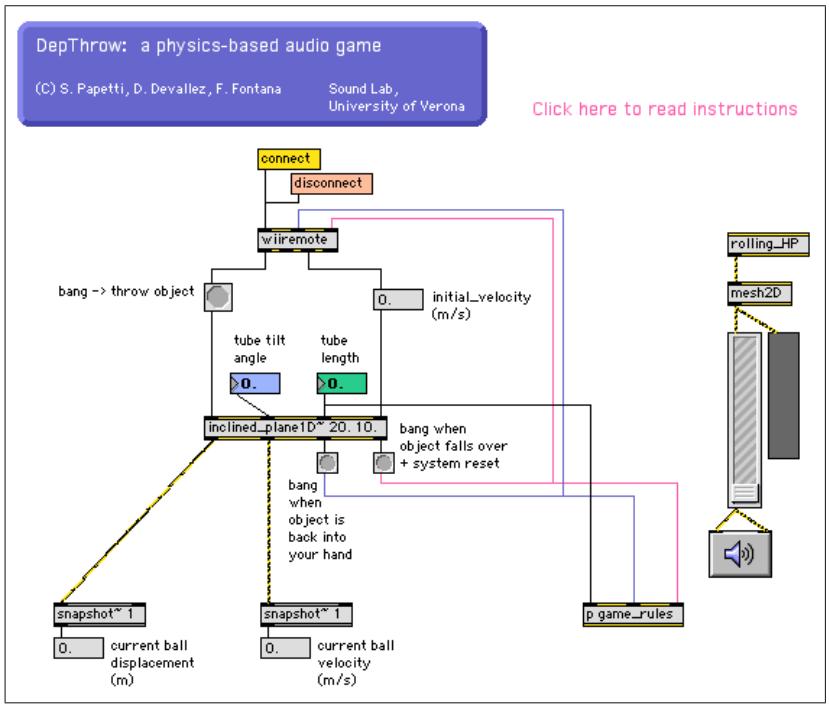


Fig. 3.6. The main Max/MSP patch implementing DepThrow.

3.2.2 Components

The first (topmost) model is a simplified simulation of the dynamics of an inclined plane. It is currently possible to set the length and tilt-angle of the plane. The model takes as input the initial velocity of an object moving upward along the plane, and returns its current displacement and velocity which are influenced by gravity. As for the implementation, it was chosen to return these quantities as signals (in contrast to controls, see Section 2.2.1) in order to allow high-rate control of the subsequent models. For this reason, pre-made dynamics libraries such as *pmpd* [13] were discarded, and a dynamics simulation has been developed as a Max/MSP *external*, now included in the SDT.

The sound source is represented by the rolling sound model described in Section 2.2.5. The current velocity of the virtual sounding object, which is provided by the inclined plane model, is used to drive the rolling model.

The spatialization process make use of the model described in Section 2.2.4. The current position of the virtual sounding object, which is provided by the inclined plane model, is used to drive the spatialization model.

[13] `http://drpichon.free.fr/pmpd/`

3.2.3 Objectives and issues

DepThrow has been added to the SDT as an advanced Max/MSP example patch. The only requirements needed to be able to use it without modifications, are a Wii Remote controller and a computer with Bluetooth connectivity running Max/MSP (even the free runtime version). The user can either mount her/his Wii Remote on a toy trolley as in the original prototype, or simply move the controller horizontally along its longitudinal axis in order to throw the virtual trolley.

Besides testing whether the game is entertaining, the user performance allows to evaluate the playability, which is partly the result of a perception-compliant mapping of the user's gesture (the act of throwing the virtual object), that is the mapping of the acceleration signals sensed by the Wii Remote onto the control parameters of the environment dynamics (inclined plane) and the rolling model. Such adequate mapping was achieved through iterative development and informal evaluation of the prototype.

An important issue pointed out by DepThrow, is whether users are able to correctly perceive the acoustic rendering of the far end of the virtual tube in order to avoid the virtual trolley falling over. Several researches have pointed out that human ability to perceive distance is relatively weak in comparison with the perception of direction [121]. The most problematic issue during listening experiments on distance perception is the difficulty for humans of separating the distance cues from the spectral characteristics of the sound source itself. As a result, users may need some *a priori* information about the sound source, or a preliminary phase of training. Moreover, many studies (see for example [43]) agree on the systematic deviation of the perceived distance of the sound source from its physical (actual) distance, and on high response variability across listeners. In particular, distance perception is very poor without reverberation. However, a recent study [100] has shown that distance accuracy improves with experience in reverberant environments, thus suggesting that the user performance may improve with training. User testing should be carried out to validate these assumptions.

Of interest are also the studies by Yao and Hayward [120] and Rath and Rocchesso [94]. In [120] the authors studied the cues exploited by users to locate the position of a virtual rolling or sliding ball inside a tube, and showed the relatively good ability of users to guess the length of the tube just by tilting it and perceiving (from auditory and/or haptic cues) the virtual object's dynamics. A physically-based model of rolling sound was also used in [94] as a continuous auditory feedback for balancing a ball along a tiltable track. To evaluate the interface, subjects had to move a virtual ball until a specific position on the track. Even without any visual display, all subjects managed to fulfill the task, meaning that people were able to locate the virtual ball making only use of auditory feedback and proprioception.

3.3 Niw Shoes

A prototype of a wearable shoe-based interface has been developed [84], which provides the user with audio-haptic cues of ground. Data supplied by sensors embedded in the insoles drive a set of physically-based synthesis models from the

SDT running on a laptop, which in turn provide their resulting output to a pair of shoe-mounted loudspeakers and to vibration actuators embedded inside the soles. By informing the synthesis models with ecological properties of grounds, neutral floors can be interactively augmented so as to react like they were made of a different material.

This prototype is one recent outcome of the EU-funded project *NIW*, which is reviewed in the introduction to this thesis.

Among the interactive experiences that have been proposed in literature to convey information through non-visual modalities, those which are mediated by the feet are still a negligible minority. Yet feet are, along with the hands, the most accurate tactile sensors we have in our external body.

Current state-of-the-art reactive floors [117] are mostly visual: some examples are provided by Ada's luminous tactile interactive floor [24] and Nintendo's prototyped floor screen.[14]

Augmented ground interactions based on auditory and tactile feedback represent an alternative to current trends: by relieving the user from looking down to the floor, foot-based interfaces would afford a rich exchange of information taking place during everyday foot-based tasks, enhancing the modalities in which the environment can be experienced.

A closed-loop foot-ground interaction paradigm is proposed with the Niw Shoes, which relies neither on vision nor on traditional locomotion tasks. The basic idea is to detect the foot action of users standing on a perceptually neutral floor (that is, flat and homogeneous, such as a pavement made of concrete) and provide them with non-visual cues capable of augmenting the ground properties, so as to manipulate the ecological perception of the physical floor. Ecological cues of ground include its material (for example wood, marble, carpet, etc.) as well as its homogeneous or aggregate nature (for example, a pathway covered with gravel or snow). Another prototype application exists, which already implements this paradigm in the form of active tiles providing ecological auditory and haptic feedback in response to a force exerted over them by the foot [116]. On the other hand, the Niw Shoes aim at realizing a similar paradigm in a cost-effective wearable interface.

3.3.1 Interaction and physical design components

The shoes, which are pictured in Fig. 3.7, are able to capture foot pressure thanks to a couple of force sensors inserted in each insole: precisely, one in correspondence to the tip and one at the heel. Since the force sensors are very small, each of them is covered with a thin rigid plastic layer (see Fig. 3.8) which allows to partially decouple the sensor's area from its exact location on the insole. This trick allows to sense pressures which are distributed on slightly wider areas, corresponding to the front and back of each sole of the foot.

The four sensors are handled by a single Arduino Duemilanove[15] acquisition board, which acts as an A/D converter. Also, the board is programmed to perform a rough noise canceling on the acquired data, namely by filtering out small changes

[14] `http://video.google.com/videoplay?docid=-6185946983584477029`.
[15] `http://www.arduino.cc/`

(a) A previous version of the shoes.

(b) The latest version of the interface.

Fig. 3.7. Two versions of the shoes prototypes. The latest prototype, shown in Fig. 3.7(b), embeds a couple of vibration actuators per shoe, while previous versions as that of Fig. 3.7(a) did not include them. Another major improvement of the latest version is that all signals from and to the shoes have been gathered together using a RS-232 serial cable, thus making the interface more robust, uncluttered and easy to wear.

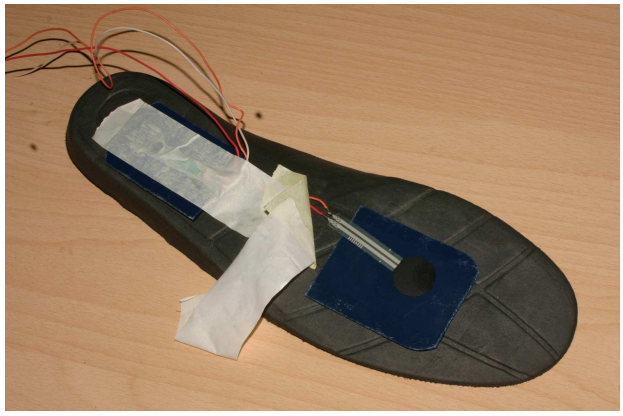


Fig. 3.8. The insoles with two sensors applied, respectively in correspondence to the tip and the heel. Each sensor is covered with a thin rigid plastic layer which allows to decouple the sensor's area from its exact location on the insole.

in the inputs. Such data are finally sent to a laptop via USB connection, which furthermore powers the Arduino. In more detail, the output from the Arduino drives a set of models from the SDT, which are capable of describing different foot-ground interaction events in terms of their physical dynamics.

The laptop is in charge of performing the overall numerical processing, providing as a result signals that feed both the vibration actuators and loudspeakers.

Loudspeakers that are able to reproduce low frequencies usually need to have a certain size and weight, and their installation on a pair of shoes is therefore unacceptable. On the other hand, keeping such loudspeakers far from the actual place of interaction would invalidate a correct sound localization, and consequently a realistic auditory experience. It is known by acousticians that high frequencies are mainly responsible for the sound source localization inside rooms, the low

frequencies rather contributing to create a stationary sound field [106]. In order to solve this problem it has been chosen to provide audio feedback in two ways:

1. via a tiny USB mobile audio interface (ESI UGM-96) which is connected to a battery-operated audio amplifier. The amplified signals are then provided to a couple of small and lightweight battery-operated loudspeakers which are attached on top of each shoe, in correspondence to the instep. As anticipated, those loudspeakers are only able to provide mid and high frequency content;
2. via an efficient and lightweight wide-band loudspeaker (Yamaha NX-U20) which is directly fed and powered via USB. This loudspeaker is in charge of providing mid and low frequency content, and is carried by the user in a backpack.

Indeed, this design solution has a clear physical interpretation since – due to their size – the shoes cannot produce low frequencies. Conversely, the floor can resonate in the low frequency range if a sufficiently large area of it oscillates in the neighborhood of the foot-floor contact position (as in, for example, hollow wooden floors).

Aside from the wide-range loudspeaker, the backpack also hosts the laptop, along with the amplifier and the acquisition board (see Fig. 3.9).

Synthetic feedback in a multimodal augmented reality context gives rise to a credible and unitary perception setting only if displayed no later than 10 ms after the real event [2]. Holding this remark, it was chosen to minimize every possible source of temporal latency, therefore avoiding standard digital wireless connections (as, for example, Bluetooth or Wi-Fi) from and to the shoes.

On the other hand, while wearability requirements impose technological restrictions and actuated shoes should avoid interferences and afford normal foot activity, the use of wired connections potentially constitutes a hindrance to the natural movement of feet and legs.

3.3.2 Real-time feedback

In the last few years, physically-based models have progressively gained popularity in the field of non-visual human-computer interaction with everyday objects [96]. An example of ecological-founded synthesis of walking sound is described in [19].

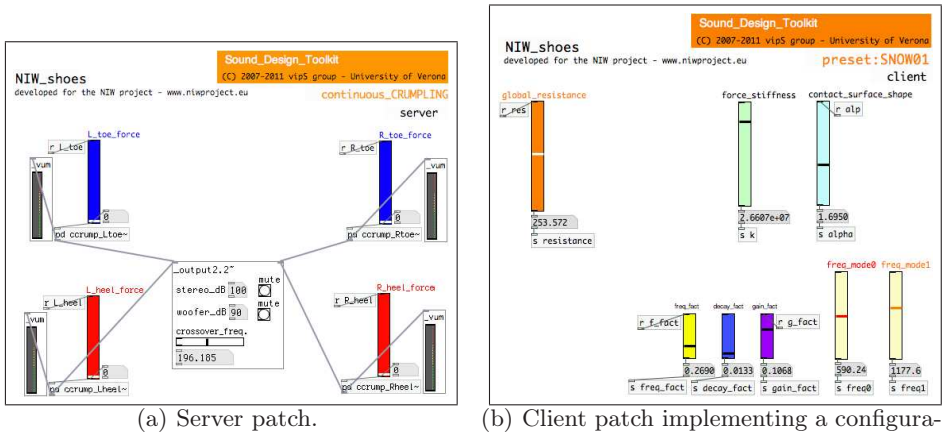
Once their parameters are properly tuned, physically-based sound models can simulate for example a displacement signal in response to an excitation, and this signal can be interpreted as a source of air pressure perturbation producing sound, as well as of floor vibrations for use in shoe-based haptic actuators.

The SDT's *continuous-crumpling* model (see Section 2.2.5) – which has already been exploited for synthesizing credible sounds and haptic feedback in simulations of walking on aggregate grounds [15, 116] – was found particularly suitable for experimentation. Figure 3.10 shows the patches for Max/MSP which have been used in connection with the Niw Shoes for interactively synthesizing audio-haptic cues of aggregate grounds. In the perspective of simulating a virtual aggregate ground, the resistance parameter allows control of the compactness of the ground (the lower the resistance, the smoother and more uniform the sequence of crackling events), while the force parameter, being proportional to the energy of the micro-impact events, can be mapped directly to the pressure exerted by the foot on

Fig. 3.9. The complete interface, with the early version shoes prototype. The backpack hosts a laptop and an acquisition board which senses data from the shoes. These data drive a set of real-time physically-based models running on the laptop and providing low-latency audio feedback through the shoe-mounted speakers.

the ground. By considering that aggregate grounds dynamically respond to a foot falling on them or scraping over them, it was hypothesized that the model should emit energy proportionally to the force changes coming from the force sensors, accounting for corresponding variations of the foot compression. Therefore it was chosen to drive the force parameter using the force variations values coming from the sensors. In particular, gate functions were employed, which filter out negative variations of the force, thus excluding feedback when the foot depresses the ground.

Homogeneous floors have also been object of simulations: in this case the aim was to add resonances to neutral floors in order to change the perceived ground material. For instance, a floor made of concrete can be augmented so as to resonate and vibrate like a wooden floor. To this end, the *soft impact* model (see Section 2.2.5) have been considered, where a resonating object is excited by noise bursts triggered by the sensors. Figure 2.7 shows the client patch for Max/MSP which has been used in connection with the Niw Shoes for interactively synthesizing audio-haptic cues of homogeneous floors. The rationale behind this idea is that noise can describe to some extent the tight sequence of microscopic impacts resulting from foot-floor interaction distributed along space and time. The force signals provided by the sensors have been used to shape the amplitude of



(a) Server patch.

(b) Client patch implementing a configuration of parameters which allow to simulate footsteps on snow.

Fig. 3.10. Patches for Pd which use the *continuous-crumpling* model in order to render audio-haptic cues of footsteps on aggregate grounds.

the noisy excitation along time: the larger this amplitude, the more energetic the micro-impacts.

3.3.3 Goals and future work

The potential advantages of wearing actuated shoes are manifold. Scenarios where they could be profitably used include navigation aids to normally gifted as well as visually impaired people, for instance along functional spaces such as airports and railway stations; auditory and haptic interventions for manual work in sensory deprived settings (for example astronautic/undersea labor, or teleoperation); rehabilitation of persons affected by foot-level proprioceptive disorders. In all such application scenarios, the provision of non visual multimodal floor cues, that are ecological rather than encoded into some higher-level semantics, would enable to use the interface in diverse cultures and social systems, as well as facilitate the learning and acceptance of the interactions. Finally, actuated shoes may contribute to improve the immersion in virtual environments, for both simulation and gaming/entertainment purposes.

While the shoes prototype still needs to undergo a proper evaluation process, at this stage we can claim that some preliminary informal tests reported positive and sometimes enthusiastic user feedback, while other users were instead more skeptical about the possibility of making profitable use of the interface.

A systematic phenomenology of foot-floor interactions does not exist yet: apart from some pioneering explorations, the psychophysics of foot-floor interactions is still unknown to a wide extent, especially from a quantitative point of view [52]. Should the proposed prototypes be of help in improving upon this lack of knowledge – for instance by providing opportunities for designing experiments with good

level of control – then augmented shoes may ultimately find their place among the components that today are employed in multimodal augmented reality setups.

Conclusion

Several studies have been carried out in recent years, dealing in different ways with issues related to sound feedback perception in, among others, product design and everyday contexts. In modern human-computer interfaces, the use of sound feedback which is consistent with the interaction is rapidly taking place, and the discipline of *interactive sonification* – a subset of *auditory display* – just deals with such matter. Also, in this framework, sound is being regarded as a vehicle of information on different levels (e.g., functional, emotional), and not only as a veneer arbitrarily chosen by the designer, or a side-effect of a device's mechanics.

This thesis addressed problems related to the development of efficient tools for sound design in interactive sonification. In particular, the use of *physically-based sound models* has been taken into account for their expressivity and potential coherence with the user's gesture.

The choice of the acoustic phenomena to be simulated has fallen on *everyday sounds* for their natural and meaningful character. Indeed, they allow to exploit a "language" which is universally and naturally interpreted by humans, and moreover devices sonified in this way are acoustically merged with the environment, in this respect "ecologically disappearing".

The energetically-consistent and phenomenologically-plausible behavior of physically-based models is especially crucial in real-time simulations of continuous interactions. In this regard, a nonlinear impact model has been studied using both analytical and numerical tools, with special emphasis on *energy consistency*.

Summary

The results described in this work can be summarized in the following points:

- A nonlinear physical model of impact with sound synthesis applications has been reviewed and its properties have been studied using both analytical tools and numerical simulations. In particular a novel analytical expression describing the behavior of energy as a function of the current velocity during contact is provided.

Several numerical realizations have been compared, and their shortcomings with regard to the corresponding analytical results have been pointed out, with special emphasis on energy consistency. In detail, it has been shown that, for some regions of the parameter space, the trajectories of the discretized systems may significantly drift from the analytically-derived curves.
It is demonstrated that by exploiting the analytical results provided, the inconsistencies of the numerical realizations can be amended, thus restoring the correct energy state of the simulated systems, as well as their stability and accuracy, during and after contact.
Finally, the computational cost and accuracy of simulations with and without corrections are compared and evaluated, showing that it is possible to obtain very accurate, stable yet efficient numerical systems.

- The development and structure of a software product for ecologically-founded sound synthesis and design is described in detail: the Sound Design Toolkit (SDT) offers state-of-the-art physically-consistent tools for designing, synthesizing and manipulating ecological sounds in real-time. The aim of the SDT is to provide efficient and effective instruments for interactive sonification and sonic interaction design.
  The SDT is cross-platform and runs natively in Max/MSP and Pd; nonetheless, the included source code forms a library of physically-based/-informed/-inspired algorithms for sound synthesis and control, which can be re-used potentially in any other environment. For easier development, a SVN repository has been set up, which is publicly accessible.
  A detailed user manual has been written, which shows the package content and functionality, with particular attention to the description and operation of the algorithms provided.
  Recently, several new algorithms have been added, together with a help system and a growing collection of advanced examples, presets (that is, recallable sets of parameters for the sound models) and tools which enable to exploit the models to their full potential, showing the wide and expressive sound palette achievable with the SDT.
  The aim is to make the SDT an ever more accessible and flexible tool for sound designers, facilitating its use in real-time interactive applications.
- Some prototype interactive applications have been reviewed, which were recently developed exploiting the tools offered by the SDT towards sonic interaction design and interactive sonification.
  - In particular, the Gamelunch is a sonically augmented dining table, whose aim is to let users experience the importance of interaction-coherent sounds in everyday-life acts. Starting from the observation that in our everyday experience, interactions and sound feedback are linked by a marked causality, it is suggested that the use of artificial sound feedback can either emphasize or contradict such causality. In this regard, the Gamelunch offers a way to investigate the loop "interaction-sound-emotion", in a way exploiting the principle of contradiction: while performing usual dining actions the user encounters contradicting and unexpected sound feedbacks, thus experiencing *per absurdum* the importance of interaction-consistent sound in everyday-life acts.

- DepThrow is an interactive audio-only game in which performance is closely related to the user's ability to perceive the dynamic distance of a virtual sounding object. Therefore the game represents a potential tool for exploring the usability of auditory distance information in interaction design.
- The Niw Shoes are a wearable shoe-based prototype interface which provides the user with audio-haptic cues of ground, in such a way that neutral floors can be interactively augmented so as to react like they were made of a different material. The potential advantages of wearing actuated shoes are manifold: some example scenarios where they could be profitably used include augmented and virtual realities, navigation aids and rehabilitation.

Appendix: Ongoing work

As I had to learn during my doctoral period, while doing research one often undertakes dead-ends, or is not be able to keep up with all the ideas he/she has had. As a consequence, a researcher inherently "wastes" precious time, either coming to negative results (most probably deriving from wrong assumptions), or not finding a way to make her/his ideas real. Besides, sometimes it happens that one has supposedly worthwhile ideas indeed, but she/he cannot find the time to investigate them thoroughly.

In this regard, the following Appendix A illustrates some early-stage results of a study that has been started with the aim of developing a more accurate and stable modal object, which would substitute that already included as part of the SDT's library (see Section 2.2.4). What is fundamentally still missing, is a validation of the new discrete-time model and corresponding implementation, in particular investigating – similarly to what is done in Chapter 1 – their energy behavior and computational cost as compared to the current realization. Nevertheless, it is worth mentioning that, from preliminary informal and qualitative evaluations, the new model seems to be far more accurate and stable, yet considerably more efficient than the current one.

Also, in reference to Chapter 1, future research will consider finding a closed-form approximation of the release velocity for the system of (1.22), where a constant external force is applied, in this way allowing to implement suitable corrections. Even without such closed-form solution, zero-finding numerical procedures could be profitably used to fix individual impacts. More interestingly for applications in acoustics, solutions will be investigated for extending the corrections suggested in Section 1.4.1 to the case where vibrational losses are present, thus being applicable to impacts with resonating objects.

A

Numerical simulations of a modal resonator

In this appendix, the continuous-time formulation of a modal resonator as a structure of parallel harmonic oscillators is examined, and its translation into a discrete-time system is shown [11, 113].

A.1 Modal resonators

A.1.1 Continuous-time description

In the SDT implementation, a modal resonator [3] model can have an arbitrary number of resonant modes, each of which is represented by a linear 2nd-order *damped harmonic oscillator* of the form:

$$\ddot{x}(t) + g\dot{x}(t) + \omega^2 x(t) = \frac{1}{m} f(t) \tag{A.1}$$

where x is the *modal displacement*, g is the mode's *damping* coefficient, ω is the *natural frequency*[1] of the mode, $1/m$ is the *modal weight*, and f is the *force* applied to the mode.

Also, the resonating object can be endowed with a macro-dynamic behavior provided by a so called *inertial mode* added to the modal resonator's structure. An inertial mode describes the macro-dynamics of a modal resonator as that of a point mass, which is described by Newton's equation of motion:

$$\ddot{x}(t) = \frac{1}{m} f(t) \tag{A.2}$$

where x is the *displacement* of the whole object, m is its *mass*, and f is the external *force* applied to the object. When present, the inertial mode is considered as the first mode of a modal resonator.

From what written above, it is clear that whereas an inertial mode is undamped, resonant modes are damped.

[1] That is, the frequency of the oscillator when no damping is present.

Having described its single components, it is now possible to describe the structure of a modal resonator having N modes of index $l = 1 \dots N$ by means of the following linear system:

$$\begin{bmatrix} \ddot{x}_1(t) \\ \vdots \\ \ddot{x}_N(t) \end{bmatrix} + \boldsymbol{G} \begin{bmatrix} \dot{x}_1(t) \\ \vdots \\ \dot{x}_N(t) \end{bmatrix} + \boldsymbol{\Omega}^2 \begin{bmatrix} x_1(t) \\ \vdots \\ x_N(t) \end{bmatrix} = \bar{\boldsymbol{m}} f(t) \tag{A.3}$$

where $\boldsymbol{G}$ and $\boldsymbol{\Omega}$ are diagonal matrices whose elements are, respectively: $g_{l=1\dots N}$ and $\omega_{l=1\dots N}$, and $\bar{\boldsymbol{m}} = [1/m_{l=1\dots N}]^T$. In case the inertial mode was present, g_1 and ω_1 would be equal to 0, while x_1 would be the displacement of the entire object and m_1 its total mass.

Taking into account N contact (or input) points of index $i = 1 \dots N$, and allowing the modal weights to vary with the position (that is, taking somewhat into account the geometry of the object), $\bar{\boldsymbol{m}}$ would be a matrix of dimensions $N \times N$ and of elements $1/m_{li}$. Moreover, an array $\boldsymbol{f} = [f_{i=1\dots N}]^T$ of separate forces applied to each interaction point would need to be considered.

On the other hand, in case the inertial mode was present, (A.3) would need to be explicitly split into one inertial and $N - 1$ resonant components: indeed, since the inertial mode behaves as a lumped point-mass representing the macrodynamics of the whole object (see (A.2)), one summing force – resulting from the overall contact forces – must applied to it.

Finally, the displacement x_j of a resonating object at a given point $j = 1 \dots N$ can be calculated as:

$$x_j(t) = \sum_{l=1}^{N} q_{jl} \cdot x_l(t) \tag{A.4}$$

where the coefficients q_{jl} are the *output weights* for each mode $l = 1 \dots N$ at the output point j. It is clear that, in case an input and an output point coincided (that is $i = j$), their modal weights $1/m_{li}$ and output weights q_{jl} would also be the same.

A.1.2 Discretization

In the current realization, the continuous-time system of (A.3) is discretized using the *bilinear transformation* (also described in Section 1.3.1). This numerical method can be seen as a conformal mapping from the Laplace-domain to the Z-domain (s-to-z):

$$s = 2F_s \frac{1 - z^{-1}}{1 + z^{-1}} \ . \tag{A.5}$$

In order to discretize the system of 2nd-order differential equations of (A.3) it is useful first to rewrite (A.1) (a single mode) as an equivalent system of two 1st-order differential equations:

$$\begin{cases} \dot{v}(t) + gv(t) + \omega^2 x(t) = \frac{1}{m} f(t) \\ v(t) = \dot{x}(t) \end{cases} \ . \tag{A.6}$$

Applying the Laplace-transform to (A.6) we obtain two 1st-order equations in s. The next step is to apply the bilinear transformation of (A.5), thus obtaining two equations in z, and finally apply the inverse Z-transform in order to obtain the following discrete-time system, expressed in *state-space* form:

$$\begin{bmatrix} x(n) \\ v(n) \end{bmatrix} = \boldsymbol{A} \begin{bmatrix} x(n-1) \\ v(n-1) \end{bmatrix} + \begin{bmatrix} \frac{1}{4m\Delta} \\ \frac{F_s}{2m\Delta} \end{bmatrix} [f(n) + f(n-1)] \tag{A.7a}$$

where the matrix $\boldsymbol{A}$ has the following expression:

$$\boldsymbol{A} = \frac{1}{\Delta} \begin{bmatrix} \Delta - \omega^2/2 & F_s \\ -F_s\omega^2 & 2F_s^2 - \Delta \end{bmatrix} \tag{A.7b}$$

and $\Delta = F_s^2 + gF_s/2 + \omega^2/4$.

As for the inertial mode, the discrete counterpart to (A.2) is easily obtained from (A.7a) and (A.7b) considering $g = 0$ and $\omega = 0$. It follows that the bilinear transformation enables to maintain a unified formulation for both the inertial and resonant modes.

Since the bilinear transformation is an implicit method, the resulting discrete (A.7a) is also in implicit form, that is to say an instantaneous dependency between the state variables (displacement x and velocity v) and the input force f is present.

Finally, the discrete-time counterpart to the system made of (A.3) and (A.4) – that is representing an entire modal object – can be written as:

$$\begin{cases} \begin{bmatrix} x_l(n) \\ v_l(n) \end{bmatrix} = \boldsymbol{A}_l \begin{bmatrix} x_l(n-1) \\ v_l(n-1) \end{bmatrix} + \begin{bmatrix} \frac{1}{4m_l\Delta_l} \\ \frac{f_s}{2m_l\Delta_l} \end{bmatrix} [f(n) + f(n-1)] \\ \\ \begin{bmatrix} x_j(n) \\ v_j(n) \end{bmatrix} = \sum_{l=1}^{N} q_{jl} \begin{bmatrix} x_l(n) \\ v_l(n) \end{bmatrix} \end{cases} \tag{A.8}$$

where $l = 1 \dots N$ and $j = 1 \dots N$ indicate respectively the mode and output point considered. The matrix $\boldsymbol{A}_l$ is as in (A.7b), but now accounts for a different $\Delta_l = F_s^2 + g_l F_s/2 + \omega_l^2/4$ for each mode l.

In order for (A.8) to handle the inertial mode or multiple input points, it is enough to follow the remarks pointed up for the continuous-time case in the previous section.

Issues

The bilinear transformation is not affected by *aliasing*, since it maps the entire imaginary axis of the s-plane onto the unit-circle in the z-plane. However, this comes at the cost of *frequency warping*: that is the frequency axis gets distorted in such a way that the correspondence between the continuous and the discrete frequencies follows the nonlinear mapping [77]:

$$\omega_{BT} = 2F_s \arctan(\frac{\omega}{2F_s}) \ . \tag{A.9}$$

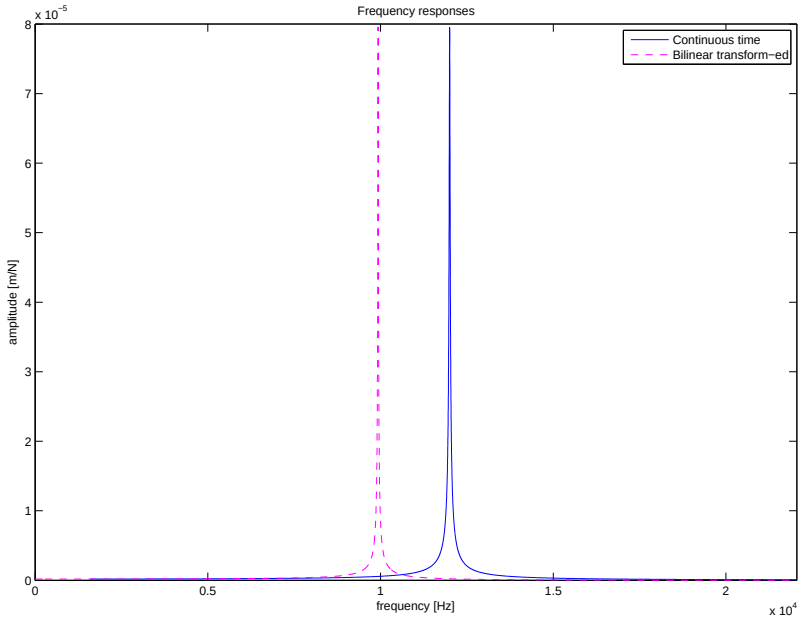


Fig. A.1. Frequency responses of a continuous-time and bilinear transform-ed harmonic oscillator having natural frequency $\omega = 12000$ Hz, damping coefficient $g = 166.6667$ Hz and modal mass $m = 0.001$ Kg. The sampling frequency is $F_s = 44.1$ kHz. The effect of warping is clearly evident: the frequency of the discretized oscillator is $\omega_{BT} = 9928.78$ Hz.

From the equation above it is clear that such warping gets more marked as the frequencies come near to the Nyquist frequency $F_s/2$, nevertheless – as Fig. A.1 shows – it can be considerable even for comparatively lower frequencies. In order to compensate for such distortion, *pre-warping* techniques should be taken into account, that is the continuous-time frequencies are pre-warped according to (A.9) prior to discretizing the system.

Moreover, as its frequency increases, the discretized harmonic oscillator results in being under-damped compared to its continuous counterpart (that is, $g_{BT} < g$), while the initial amplitude becomes smaller (that is, the oscillator initially absorbs more energy, which implies that $m_{BT} > m$). Figure A.2 shows a comparison between the impulse responses of a continuous-time oscillator and that of its discrete-time counterpart.

A.2 Study of the harmonic oscillator

A.2.1 Analytic time-domain solution

In order to better analyze the harmonic oscillator, one could start looking for the analytic solution of the homogeneous equation corresponding to (A.1) [11, 113]:

$$\ddot{x}(t) + g\dot{x}(t) + \omega^2 x(t) = 0 \; . \tag{A.10}$$

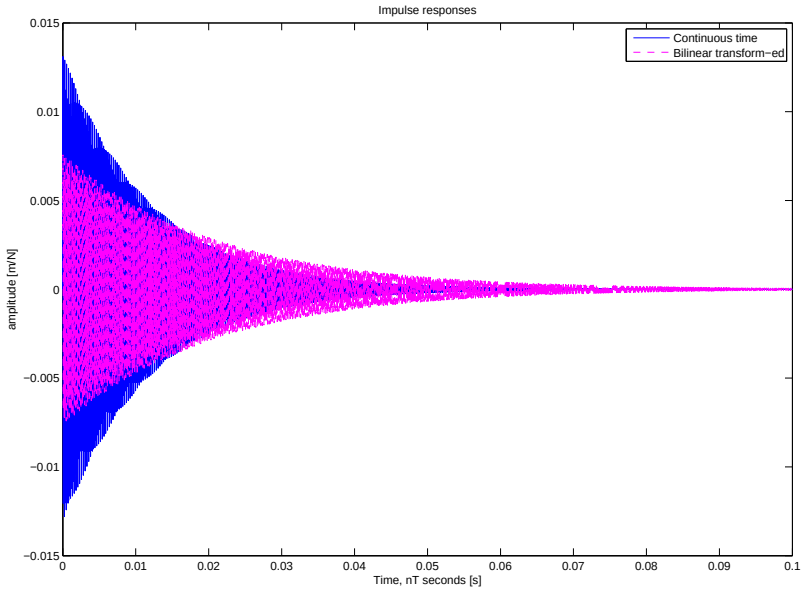


Fig. A.2. Impulse responses of a continuous-time and the bilinear transform-ed harmonic oscillator having natural frequency $\omega = 12000$ Hz, damping coefficient $g = 166.6667$ Hz and modal mass $m = 0.001$ Kg. The sampling frequency is $F_s = 44.1$ kHz. It is clear that the damping g_{BT} and the modal mass m_{BT} of the discretized oscillator are respectively lower and greater than the original continuous time versions g and m_{BT}.

In more detail, the general solution can be found starting from the corresponding characteristic equation:

$$\lambda^2 + g\lambda + \omega^2 = 0 \tag{A.11}$$

which has the following roots:

$$\lambda_{1,2} = \frac{-g \pm \sqrt{g^2 - 4\omega^2}}{2} = -\frac{g}{2} \pm \sqrt{\left(\frac{g}{2}\right)^2 - \omega^2} \ . \tag{A.12}$$

Depending on the roots' values, the general solution may assume different forms. For the purpose of simulating the normal modes of resonance of a sounding object, cases of interest are represented by configurations where the damping is sufficiently small compared to the natural frequency of oscillation[2]:

$$\left(\frac{g}{2}\right)^2 < \omega^2 \ . \tag{A.13}$$

Therefore, the corresponding complex-valued roots are:

$$\lambda_{1,2} = -\frac{g}{2} \pm j\bar{\omega} \tag{A.14}$$

[2] In that case the oscillator is *under-damped* or, for $g = 0$, *not damped*. Other possible configurations, which are not of interest, are: *critical damping*, for $\left(\frac{g}{2}\right)^2 = \omega^2$, and *overdamping*, for $\left(\frac{g}{2}\right)^2 > \omega^2$.

where:

$$\bar{\omega} = \sqrt{\omega^2 - \left(\frac{g}{2}\right)^2} \,. \tag{A.15}$$

Moreover, if g is negligible compared to ω, that is for $(g/2)^2 \ll \omega^2$, the approximation $\bar{\omega} \approx \omega$ holds.

We can now find the general solution of (A.10) which is:

$$x(t) = C_1 e^{(-\frac{g}{2} + j\bar{\omega})t} + C_2 e^{(-\frac{g}{2} - j\bar{\omega})t} \tag{A.16}$$

where C_1 and C_2 are complex-valued. Only the real part of this solution has a physical meaning, and it can be shown that the real-valued solution is always an exponentially decaying sinusoidal wave of the form:

$$x(t) = Ce^{-\frac{g}{2}t} \cos(\bar{\omega}t - \phi) \tag{A.17}$$

where C and ϕ depend on both the initial conditions and the driving force.

A.2.2 Filter interpretation

Considering the harmonic oscillator as an analog filter, and applying the Laplace transform to (A.1), its *transfer function* can be written as:

$$H(s) = \frac{F(s)}{X(s)} = \frac{1}{m} \cdot \frac{1}{s^2 + 2\frac{g}{2}s + \omega^2} \,. \tag{A.18}$$

Also, the continuous-time impulse response of (A.1) can be computed by using the inverse Laplace transform. In particular, if the assumption of (A.13) holds, the transfer function (A.18) has complex poles, and the corresponding *impulse response* is:

$$h(t) = \frac{1}{m\bar{\omega}} \cdot e^{-\frac{g}{2}t} \cdot \sin(\bar{\omega}t) = \frac{1}{m\bar{\omega}} \cdot e^{-\frac{t}{\tau}} \cdot \sin(\bar{\omega}t) \tag{A.19}$$

where $\bar{\omega}$ is defined as in (A.15) and $\tau = 2/g$ is the *decay time*, defined as the time taken to reach an amplitude equal to $1/e$ of the initial oscillation amplitude. It is self-evident that for $f(t) = \delta(t)$ and initial conditions $x(0) = \dot{x}(0) = 0$, (A.17) is equivalent to (A.19); that is, it describes the impulse response of the harmonic oscillator.

A.2.3 Discretization

Considering the above observations and results, one could note that instead of directly discretizing (A.1) – as done in Section A.1.2 and [8] – it is alternatively and equivalently possible to discretize its impulse response given in (A.19). To this end, provided that the decay time τ is not too short,[3] (A.19) can be considered as a band-limited signal, and it is therefore possible to discretize it using the

[3] Indeed, only extremely short decay times are excluded, which are of no interest for a *resonant* mode. Moreover, since (A.19) assumes that (A.13) holds, the decay time is guaranteed to be "not too short".

straightforward *impulse-invariant transformation* [77, 85, 108]. In order to avoid noticeable artifacts due to *aliasing* we have to take care of not implementing any modes having frequency which is near or above the Nyquist frequency $Fs/2$. Once solved the aliasing issue, the impulse-invariant transformation appears to be an optimal option. In fact, since a pole p of an analog filter is mapped to the pole $p_{II} = e^{p/F_s}$ of its digital counterpart, stability is guaranteed at any sample rate.[4]

The discretized version of (A.19) is therefore:

$$h(n) = T \cdot h(nT) = \frac{1}{F_s} \cdot h\left(\frac{n}{F_s}\right) = \frac{1}{m\bar{\omega}F_s} \cdot e^{-\frac{n}{\tau F_s}} \cdot \sin\left(\bar{\omega}\frac{n}{\tau F_s}\right) \tag{A.20}$$

where the sampling interval T is used as a scaling coefficient, since the discrete-time Dirac impulse $\delta(t_n)$ has an area of T, while the continuous-time Dirac impulse $\delta(t)$ has unit area.

By applying the Z-transform to the discrete-time impulse response given in (A.20), the corresponding discrete-time transfer function is obtained. In order to easily apply the Z-transform, (A.20) can be rewritten as:

$$\begin{aligned} h(n) &= \frac{1}{m\bar{\omega}F_s} \cdot e^{-\frac{n}{\tau F_s}} \cdot \frac{e^{\frac{j\bar{\omega}n}{F_s}} - e^{-\frac{j\bar{\omega}n}{F_s}}}{2j} = \\ &= \frac{1}{2jm\bar{\omega}F_s} \cdot \left[(e^{-\frac{1}{\tau F_s}} e^{\frac{j\bar{\omega}}{F_s}})^n - (e^{-\frac{1}{\tau F_s}} e^{-\frac{j\bar{\omega}}{F_s}})^n\right] . \end{aligned} \tag{A.21}$$

Finally, the corresponding discrete-time transfer function is obtained as:

$$\begin{aligned} H(z) &= \underbrace{\frac{1}{2jm\bar{\omega}F_s}}_{G} \cdot \left(\frac{1}{1 - \underbrace{(e^{-\frac{1}{\tau F_s}} e^{\frac{j\bar{\omega}}{F_s}})}_{p} z^{-1}} - \frac{1}{1 - \underbrace{(e^{-\frac{1}{\tau F_s}} e^{-\frac{j\bar{\omega}}{F_s}})}_{p^*} z^{-1}} \right) = \\ &= G \cdot \left(\frac{1}{1 - pz^{-1}} - \frac{1}{1 - p^* z^{-1}} \right) = G \cdot \frac{(p - p^*)z^{-1}}{1 - pz^{-1} - p^* z^{-1} + |p|^2 z^{-2}} = \\ &= \frac{2jG\Im\{p\}z^{-1}}{1 - 2\Re\{p\}z^{-1} + |p|^2 z^{-2}} \end{aligned} \tag{A.22}$$

where the asterisk $*$ indicates complex conjugation. For the sake of clarity, the transfer function of (A.22) is expressed in a more general form hereafter:

$$H(z) = \frac{b_1 z^{-1}}{1 + a_1 z^{-1} + a_2 z^{-2}} , \tag{A.23a}$$

$$b_1 = 2jG\Im\{p\} , \tag{A.23b}$$

$$a_1 = -2\Re\{p\} , \tag{A.23c}$$

$$a_2 = |p|^2 . \tag{A.23d}$$

[4] More precisely, poles in the Z-domain are stable (that is, they reside within the unit circle: $|p_{II}| < 1$) if and only if the corresponding poles in the Laplace-domain are stable (that is, they have negative real-part: $\Re\{p\} < 0$).

From the above results, it is evident that each resonant mode of a modal resonator can be implemented as a "complex-pole no-zero" digital filter. The equation expressing the displacement of a resonant mode hence is:

$$\begin{aligned} x(n) &= -a_1 x(n-1) - a_2 x(n-2) + b_1 f(n-1) = \\ &= 2\Re\{p\}x(n-1) - |p|^2 x(n-2) + 2jG\Im\{p\}f(n-1) \end{aligned} \tag{A.24}$$

where it should be noted that in order to compute the displacement x at the time instant n numerically, we only need the force f at the instant $n-1$. In other words, (A.24) is in explicit form.

It is worth noticing that – differently from what has been done in Section A.1.2 – the discretization by means of the impulse-invariant transformation does not allow to handle both the inertial and resonant modes in a uniform way. On the other hand, the discretization of the equation of motion for a point-mass (which is equivalent to the inertial mode) is thoroughly covered in Section 1.3.1. Such results can be exploited together with those derived in this section, in order to provide an alternative discrete-time model of the modal resonator as described in Section A.1.

A.2.4 Remarks on the implementation efficiency

With the purpose of improving the efficiency of software implementations corresponding to the results provided in this section, it is desirable to rewrite the coefficients of (A.24). As a first step, according to (A.22) a pole p can be expressed as follows:

$$p = e^{-\frac{1}{\tau F_s}} e^{\frac{j\bar{\omega}}{F_s}} = e^{-\frac{1}{\tau F_s}} \left[\cos\left(\frac{\bar{\omega}}{F_s}\right) + j \sin\left(\frac{\bar{\omega}}{F_s}\right) \right] . \tag{A.25}$$

Then, the quantities appearing in (A.24) which make use of p can be rewritten as [122]:

$$\Re\{p\} = e^{-\frac{1}{\tau F_s}} \cos\left(\frac{\bar{\omega}}{F_s}\right) , \tag{A.26a}$$

$$\Im\{p\} = e^{-\frac{1}{\tau F_s}} \sin\left(\frac{\bar{\omega}}{F_s}\right) , \tag{A.26b}$$

$$|p| = e^{-\frac{1}{\tau F_s}} . \tag{A.26c}$$

Further, it can be noted that the complex coefficient G only appears in (A.23b) where it multiplies $2j$, giving the following real quantity as result:

$$2jG = \frac{1}{m\bar{\omega}F_s} . \tag{A.27}$$

All quantities in (A.26) and (A.27) are real-valued and can be used in actual software implementations without the need to rely on complex-arithmetics libraries.

References

1. J. Abbott and A. Okamura. Effects of position quantization and sampling rate on virtual-wall passivity. *Robotics, IEEE Transactions on*, 21(5):952–964, Oct. 2005.
2. B. D. Adelstein, D. Begault, M. Anderson, and E. Wenzel. Sensitivity to haptic-audio asynchrony. In *Proc. Int. Conf. on Multimodal Interfaces (ICMI '03)*, pages 73–76, New York, NY, USA, 2003. ACM.
3. J. M. Adrien. The missing link: Modal synthesis. In G. De Poli, A. Piccialli, and C. Roads, editors, *Representations of Musical Signals*, pages 269–297. MIT Press, Cambridge, MA, 1991.
4. G. Anceschi. Basic design, fondamenta del design. In M. A. Garito, G. Anceschi, and M. Botta, editors, *L'Ambiente dell'Appredimento - Web Design e Processi Cognitivi*, pages 57–67. McGraw-Hill, Milan, 2006.
5. F. Avanzini and P. Crosato. Integrating physically based sound models in a multimodal rendering architecture: Research articles. *Comput. Animat. Virtual Worlds*, 17(3-4):411–419, 2006.
6. F. Avanzini, M. Rath, and D. Rocchesso. Physically-based audio rendering of contact. In *Proc. IEEE Int. Conf. on Multimedia and Expo (ICME '02)*, volume 2, pages 445–448, 2002.
7. F. Avanzini and D. Rocchesso. Modeling collision sounds: non-linear contact force. In *Proc. Int. Conf. on Digital Audio Effects (DAFx '01)*, pages 61–66, Limerick, December 2001.
8. F. Avanzini and D. Rocchesso. Efficiency, accuracy, and stability issues in discrete-time simulations of single reed instruments. *J. of the Acoustical Society of America*, 111(5), 2002.
9. F. Avanzini and D. Rocchesso. Physical modeling of impacts: theory and experiments on contact time and spectral centroid. In *Proc. Int. Conf. on Sound and Music Computing (SMC '04)*, pages 287–293, 2004.
10. F. Avanzini, D. Rocchesso, and S. Serafin. Modeling interactions between rubbed dry surfaces using an elasto-plastic friction model. In *Proc. Int. Conf. on Digital Audio Effects (DAFx '02)*, Hamburg, Germany, 2002.
11. B. Bank. *Physics-based Sound Synthesis of String Instruments Including Geometric Nonlinearities*. PhD thesis, Department of Measurement and Information Systems, Budapest University of Technology and Economics, 2006.
12. S. Bilbao. Robust physical modeling sound synthesis for nonlinear systems. *Signal Processing Magazine, IEEE*, 24(2):32–41, March 2007.
13. S. Bilbao. *Numerical Sound Synthesis*. Wiley, New York, USA, 2009.

14. G. Borin, G. De Poli, and D. Rocchesso. Elimination of Delay-Free Loops in Discrete-Time Models of Nonlinear Acoustic Systems. *IEEE Trans. on Speech and Audio Processing*, 8(5):597–605, September 2000.
15. R. Bresin, S. Delle Monache, F. Fontana, S. Papetti, P. Polotti, and Y. Visell. Auditory feedback through continuous control of crumpling sound synthesis. In *Sonic Interaction Design: a Conf. on Human Factors in Computing Systems (CHI '08) Workshop*, Florence, Italy, 2008. COST Action IC061, ACM/SIGCHI.
16. A. Chaigne and J. Kergomard. *Acoustique des Instruments de Musique*. Belin, 2008.
17. J. E. Colgate and J. M. Brown. Factors Affecting the Z-Width of a Haptic Display. In *Proc. IEEE Int. Conf. on Robotics & Automation*, pages 3205–3210, San Diego, May 1994.
18. P. R. Cook. Physically inspired sonic modeling (PhISM): Synthesis of percussive sounds. *Computer Music Journal*, 21(3):38–49, 1997.
19. P. R. Cook. Modeling Bill's gait: analysis and parametric synthesis of walking sounds. In *AES 22nd Int. Conf. on Virtual, Synthetic and Entertainment Audio*, pages 73–78, 2002.
20. P. R. Cook. *Real Sound Synthesis for Interactive Applications*. A. K. Peters, Ltd., Natick, MA, USA, 2002.
21. P. R. Cook and G. P. Scavone. The Synthesis ToolKit (STK). In *Proc. Int. Computer Music Conf. (ICMC '99)*, Beijing, China, 1999.
22. A. Crevoisier and P. Polotti. Tangible acoustic interfaces and their applications for the design of new musical instruments. In *Proc. conf. on New Interfaces for Musical Expression (NIME '05)*, pages 97–100, Singapore, Singapore, 2005. National University of Singapore.
23. G. De Poli and D. Rocchesso. Physically-based Sound Modelling. *Organised Sound*, 3(1):61–76, 1998.
24. T. Delbrück, A. M. Whatley, R. Douglas, K. Eng, K. Hepp, and P. Verschure. A tactile luminous floor for an interactive autonomous space. *Robotics and Autonomous Systems*, 55(6):433–443, June 2007.
25. S. Delle Monache. Personal communication, 2008.
26. S. Delle Monache and P. Polotti. Gamelunch - the sonic dining. In *Enaction in Arts Catalogue (Enactive '07)*, page 25, 2007.
27. S. Delle Monache, P. Polotti, S. Papetti, and D. Rocchesso. Gamelunch, a physics-based sonic dining table. In *Proc. Int. Computer Music Conf. (ICMC '07)*, volume 2, pages 41–44, Copenhagen, Denmark, 2007.
28. S. Delle Monache, P. Polotti, S. Papetti, and D. Rocchesso. Sonically augmented found objects. In *Proc. New Interfaces for Musical Expression conf. (NIME '08)*, pages 154–157, Genova, Italy, 2008.
29. D. Devallez, F. Fontana, and D. Rocchesso. An audio-haptic interface based on auditory depth cues. In *Proc. 10th int. conf. on Multimodal interfaces (ICMI '08)*, pages 209–216, New York, NY, USA, 2008. ACM.
30. D. Devallez, F. Fontana, and D. Rocchesso. Linearizing auditory distance estimates by means of virtual acoustics. *Acta Acustica united with Acustica*, 94:813–824(12), November-December 2008.
31. N. Diolaiti. A criterion for the passivity of haptic devices. In *Int. Conf. on Robotics and Automation*, pages 2463–2468, 2005.
32. N. Diolaiti, C. Melchiorri, and S. Stramigioli. Contact impedance estimation for robotic systems. *IEEE Trans. on Robotics*, 21(5):925–935, 2005.
33. Y. Dobashi, T. Yamamoto, and T. Nishita. Real-time rendering of aerodynamic sound using sound textures based on computational fluid dynamics. *ACM Trans. Graph.*, 22(3):732–740, 2003.
34. C. Drioli. Personal communication, 2008.

35. C. Drioli and D. Rocchesso. Acoustic rendering of particle-based simulation of liquids in motion. In *Proc. Int. Conf. on Digital Audio Effects (DAFx '09)*, 2009.
36. P. Dupont, V. Hayward, B. Armstrong, and F. Altpeter. Single State Elasto-Plastic Friction Models. *IEEE Trans. on Automatic Control*, 47(5):787–792, June 2002.
37. R. Ellis, N. Sarkar, and A. Jenkins. Numerical methods for the force reflection of contact. *ASME Trans. on Dynamic Systems, Modeling, and Control,*, 119(4):768–774, Dec. 1997.
38. S. Faik and H. Witteman. Modeling of Impact Dynamics: A Literature Survey. In *Proc. Int. ADAMS User Conf.*, 2000.
39. P. Flores, J. Ambrósio, J. Claro, and H. Lankarani. Influence of the contact-impact force model on the dynamic response of multi-body systems. *Proc. Institution of Mechanical Engineers-K*, 220(1):21–34, 2006.
40. P. Flores, J. P. Claro, and H. M. Lankarani. *Kinematics and Dynamics of Multibody Systems with Imperfect Joints: Models and Case Studies*. Springer, 2008.
41. F. Fontana and R. Bresin. Physics-based sound synthesis and control: crushing, walking and running by crumpling sounds. In *Proc. XIV Colloquium on Musical Informatics (CIM '03)*, pages 109–114, Florence, Italy, 2003.
42. F. Fontana and D. Rocchesso. A physics-based approach to the presentation of acoustic depth. In *Proc. Int. Conf. on Auditory Display*, pages 79–82, Boston (MA), June 2003.
43. F. Fontana and D. Rocchesso. Auditory distance perception in an acoustic pipe. *ACM Trans. Appl. Percept.*, 5(3):1–15, 2008.
44. F. Fontana, D. Rocchesso, and E. Apollonio. Using the waveguide mesh in modelling 3d resonators. In *Proc. Conf. on Digital Audio Effects (DAFX-00)*, pages 229–232, Verona, Italy, Dec. 2000. COST-G6.
45. F. Fontana, L. Savioja, and V. Välimäki. A modified rectangular waveguide mesh structure with interpolated input and output points. In *Proc. Int. Computer Music Conf.*, pages 87–90, La Habana, Cuba, Sept. 2001. ICMA.
46. K. Franinovic, Y. Visell, and D. Hug. Sound embodied: Explorations of sonic interaction design for everyday objects in a workshop setting. In *Proc. Int. Conf. on Auditory Display (ICAD '07)*, Montreal, Canada, 2007.
47. W. W. Gaver. *Everyday listening and auditory icons*. PhD thesis, University of California, San Diego, 1988.
48. W. W. Gaver. How do we hear in the world? Explorations in ecological acoustics. *Ecological Psychology*, 5(4):285–313, 1993.
49. W. W. Gaver. Synthesizing auditory icons. In *Proc. Conf. on Human Factors in Computing Systems (INTERACT '93 and CHI '93)*, pages 228–235, New York, NY, USA, 1993. ACM.
50. W. W. Gaver. What in the world do we hear? An ecological approach to auditory event perception. *Ecological Psychology*, 5(1):1–29, 1993.
51. B. Giordano, D. Rocchesso, and S. McAdams. Integration of acoustical information in the perception of impacted sound sources: the role of information accuracy and exploitability. *Journal of Experimental Psychology: Human Perception and Performance*. In press.
52. B. L. Giordano, S. McAdams, Y. Visell, J. Cooperstock, H.-Y. Yao, and V. Hayward. Non-visual identification of walking grounds. *J. of the Acoustical Society of America*, 123(5):3412, 2008.
53. W. Goldsmith. *Impact: The Theory and Physical Behavior of Colliding Solids*. Dover Publications, New York, USA, 2nd edition, 2001.
54. J. K. Hahn, J. Geigel, J. W. Lee, L. Gritz, T. Takala, and S. Mishra. An integrated approach to motion and sound. *The J. of Visualization and Computer Animation*, 6(2):109–124, 1995.

55. V. Hayward. Physically-Based Haptic Synthesis. In M. C. Lin and M. Otaduy, editors, *Haptic Rendering: Foundations, Algorithms and Applications*, pages 297–309. A. K. Peters, Ltd., Natick, MA, USA, 2008.
56. T. Hermann and A. Hunt. Guest editors' introduction: An introduction to interactive sonification. *IEEE MultiMedia*, 12(2):20–24, 4 2005.
57. T. Hermann and H. Ritter. Model-based sonification revisited—authors' comments on Hermann and Ritter, ICAD 2002. *ACM Trans. Appl. Percept.*, 2(4):559–563, 2005.
58. R. Hoskinson, K. van den Doel, and S. Fels. Real-time adaptive control of modal synthesis. In *Proc. conf. on New Interfaces for Musical Expression (NIME '03)*, pages 99–103, Singapore, Singapore, 2003. National University of Singapore.
59. O. Houix, G. Lemaitre, N. Misdariis, P. Susini, K. Franinovic, D. Hug, J. Otten, J. Scott, Y. Visell, D. Devallez, F. Fontana, S. Papetti, P. Polotti, and D. Rocchesso. Everyday sound classification: sound perception, interaction and synthesis. Deliverable 4.1, CLOSED project, 2006.
60. P. A. Houle and J. P. Sethna. Acoustic emission from crumpling paper. *Physical Review E*, 54(1):278–283, July 1996.
61. K. H. Hunt and F. R. E. Crossley. Coefficient of restitution interpreted as damping in vibroimpact. *ASME J. Applied Mech.*, pages 440–445, June 1975.
62. J.-C. Risset. *An Introductory Catalog of Computer-synthesized Sounds*. Bell laboratories, Murray Hill, NJ, USA, 1969.
63. S. Jordá, M. Kaltenbrunner, G. Geiger, and R. Bencina. The reacTable. In *Proc. Int. Computer Music Conf. (ICMC '05)*, pages 579–582, 2005.
64. K. Kuchenbecker, J. Fiene, and G. Niemeyer. Improving contact realism through event-based haptic feedback. *IEEE Trans. on Visualization and Computer Graphics*, 12(2):219–230, March-April 2006.
65. G. Kuwabara and K. Kono. Restitution coefficient in a collision between two spheres. *Jap. J. of Appl. Phys.*, 26(8):1230–1233, 1987.
66. H. M. Lankarani and P. E. Nikravesh. A contact force model with hysteresis damping for impact analysis of multibody systems. *J. of Mechanical Design*, 112(3):369–376, 1990.
67. G. Lemaitre, O. Houix, Y. Visell, K. Franinovic, N. Misdariis, and P. Susini. Toward the design and evaluation of continuous sound in tangible interfaces: The spinotron. *International Journal of Human-Computer Studies*, 67(11):976 – 993, 2009. Special issue on Sonic Interaction Design (SID).
68. Y.-K. Lim, E. Stolterman, H. Jung, and J. Donaldson. Interaction gestalt and the design of aesthetic interactions. In *Proc. conf. on Designing pleasurable products and interfaces (DPPI '07)*, pages 239–254, New York, NY, USA, 2007. ACM.
69. M. Mahvash and V. Hayward. High-fidelity haptic synthesis of contact with deformable bodies. *IEEE Comput. Graph. Appl.*, 24(2):48–55, 2004.
70. M. Mahvash, V. Hayward, and J. Lloyd. Haptic rendering of tool contact. In *Proc. Eurohaptics*, pages 110–115, 2002.
71. D. W. Marhefka and D. E. Orin. A compliant contact model with nonlinear damping for simulation of robotic systems. *IEEE Trans. on Systems, Man, and Cybernetics - Part A: Systems and Humans*, 29(6):566–572, November 1999.
72. N. Misdariis, R. Caussé, L. Dandrel, J. Bensoam, and C. Vergez. Modalys, un outil pour le design sonore. In *1ères Journées du Design Sonore*, Paris, France, 2002.
73. M. Müller, D. Charypar, and M. Gross. Particle-based fluid simulation for interactive applications. In *Proc. ACM SIGGRAPH/Eurographics 03*, pages 154–159, 2003.
74. M. Müller, J. Stam, D. James, and N. Thürey. Real time physics: class notes. In *ACM SIGGRAPH 2008 classes*, pages 1–90, New York, NY, USA, 2008. ACM.

75. D. T. Murphy and D. M. Howard. 2-d digital waveguide mesh topologies in room acoustics modelling. In *Proc. Conf. on Digital Audio Effects (DAFX-00)*, pages 211–216, Verona, Italy, Dec. 2000. COST-G6.
76. J. F. O'Brien, P. R. Cook, and G. Essl. Synthesizing sounds from physically based motion. In *Proc. of the 28th annual conf. on Computer graphics and interactive techniques (SIGGRAPH '01)*, pages 529–536, New York, NY, USA, 2001. ACM.
77. A. V. Oppenheim, R. W. Schafer, and J. R. Buck. *Discrete-Time Signal Processing*. Prentice-Hall, Inc., Englewood Cliffs, NJ, USA, 2nd edition, 1999.
78. S. Papetti. Sintesi audio in tempo-reale mediante modelli waveguide di risonatori e modelli non-lineari di contatto. Master's thesis, 2006.
79. S. Papetti. The Sounding Object: estensione dei modelli di sintesi basati su impatto e frizione. In *Proc. XVI Colloquium on Musical Informatics (CIM '06)*, Genova, Italy, 2006.
80. S. Papetti, F. Avanzini, and D. Rocchesso. Numerical methods for a non-linear impact model: a comparative study with closed-form corrections. *IEEE Trans. Audio, Speech and Language Processing*. Submitted on Nov 10th, 2009.
81. S. Papetti, F. Avanzini, and D. Rocchesso. Energy and accuracy issues in numerical simulations of a non-linear impact model. In *Proc. Int. Conf. on Digital Audio Effects (DAFx 09)*, Como, Italy, 2009.
82. S. Papetti, D. Devallez, and F. Fontana. DepThrow: a physics-based audio game. In *Proc. 14th Int. Conf. on Auditory Display (ICAD '08)*, Paris, France, 2008.
83. S. Papetti, D. Devallez, and F. Fontana. DepThrow: uno strumento di indagine sulla percezione uditiva della distanza in forma di gioco audio. In *Proc. XVII Colloquium on Musical Informatics (CIM '08)*, Venice, Italy, 2008.
84. S. Papetti, F. Fontana, and M. Civolani. A shoe-based interface for ecological ground augmentation. In *Proc. 4th Int. Workshop on Haptic and Audio Interaction Design (HAID '09)*, volume 2, pages 28–29, Dresden, Germany, 2009.
85. T. Parks and C. Burrus. *Digital Filter Design*. Wiley, New York, USA, 1987.
86. L. Peltola, C. Erkut, P. R. Cook, and V. Välimäki. Synthesis of hand clapping sounds. *IEEE Trans. Audio, Speech and Language Processing*, 15(3):1021–1029, 2007.
87. P. Polotti, S. Delle Monache, S. Papetti, and D. Rocchesso. Gamelunch: forging a dining experience through sound. In *Proc. Conf. on Human Factors in Computing Systems (CHI '08)*, Florence, Italy, 2008. ACM/SIGCHI.
88. P.Polotti and D. Rocchesso, editors. *Sound to Sense, Sense to Sound - A state of the art in Sound and Music Computing*. Logos Verlag, Berlin, Germany, 2008.
89. L. Pust and F. Peterka. Impact oscillator with Hertz's model of contact. *Meccanica*, 38(1):99–116, 2003.
90. A. Quarteroni, R. Sacco, and F. Saleri. *Numerical Mathematics*. Springer, 2nd edition, 2007.
91. M. Rath. An expressive real-time sound model of rolling. In *Proc. Int. Conf. on Digital Audio Effects (DAFx '03)*, London, UK, September 2003.
92. M. Rath, F. Avanzini, N. Bernardini, G. Borin, F. Fontana, L. Ottaviani, and D. Rocchesso. An introductory catalog of computer-synthesized contact sounds, in real-time. In *Proc. XIV Colloquium on Musical Informatics (CIM '03)*, Florence, Italy, 2003.
93. M. Rath and F. Fontana. High level models: bouncing, breaking, rolling, crumpling, pouring. In D. Rocchesso and F. Fontana, editors, *"The Sounding Object"*. Mondo Estremo, Florence, Italy, 2003.
94. M. Rath and D. Rocchesso. Continuous sonic feedback from a rolling ball. *IEEE MultiMedia*, 12(2):60–69, 2005.
95. C. Roads. *The Computer Music Tutorial*. MIT Press, Cambridge, MA, USA, 1996.

96. D. Rocchesso, R. Bresin, and M. Fernström. Sounding objects. *IEEE MultiMedia*, 10(2):42–52, 2003.
97. D. Rocchesso and F. Fontana, editors. *The Sounding Object*. Mondo Estremo, Firenze, Italy, 2003.
98. D. Rocchesso and S. Serafin. Sonic Interaction Design. *Int. J. of Human-Computer Studies*, 67(11):905 – 906, 2009. Special issue on Sonic Interaction Design - SID.
99. D. Rocchesso, S. Serafin, F. Behrendt, N. Bernardini, R. Bresin, G. Eckel, K. Franinovic, T. Hermann, S. Pauletto, P. Susini, and Y. Visell. Sonic interaction design: sound, information and experience. In *Sonic Interaction Design: a Conf. on Human Factors in Computing Systems (CHI '08) Workshop*, pages 3969–3972, New York, NY, USA, 2008. ACM.
100. B. Shinn-Cunningham. Learning reverberation: Considerations for spatial auditory displays. In *Proc. Int. Conf. on Auditory Display (ICAD '00)*, 2000.
101. J. O. Smith. Waveguide filter tutorial. In *Proc. Int. Computer Music Conf.*, pages 9–16, Champaign-Urbana, 1987. ICMA.
102. J. O. Smith. Principles of digital waveguide models of musical instruments. In M. Kahrs and K. Brandeburg, editors, *Applications of DSP to Audio and Acoustic*, pages 417–466. Kluwer Academic Publishers, 2002.
103. J. O. Smith. *Physical Audio Signal Processing: For Virtual Musical Instruments and Digital Audio Effects*. `http://ccrma.stanford.edu/~jos/pasp/`, 2006.
104. A. Stulov. Dynamic behavior and mechanical features of wool felt. *Acta Mechanica*, 169(1):13–21, 2004.
105. P. Susini, N. Misdariis, G. Lemaitre, D. Rocchesso, P. Polotti, K. Franinovic, Y. Visell, O. K., H. Purwins, and K. Adiloglu. Synthesizing auditory icons. In *Proc. 2nd ISCA/DEGA Tutorial and Research Workshop on Perceptual Quality of Systems*, pages 228–235, 2006.
106. M. Tanaka and Y. Kaneda. Performance of Sound Source Direction Estimation Methods under Reverberant Conditions. *J. of the Acoustical Society of Japan*, 14(4):291–292, 1993.
107. D. Terzopoulos and K. Fleischer. Modeling inelastic deformation: viscolelasticity, plasticity, fracture. In *Proc. 15th annual conf. on computer graphics and interactive techniques (SIGGRAPH '88)*, pages 269–278, New York, NY, USA, 1988. ACM.
108. L. Trautmann and R. Rabenstein. Transfer function models with nonlinear excitations for digital sound synthesis. In *in X European Signal Processing Conf. (EUSIPCO 2000). IEEE*, pages 2217–2220, 2000.
109. L. Trautmann and R. Rabenstein. Stable systems for nonlinear discrete sound synthesis with the functional transformation method. In *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing (ICASSP '02)*, volume 2, pages 1861–1864, 2002.
110. K. van den Doel. Phisically-based models for liquid sounds. In *Proc. Int. Conf. on Auditory Display (ICAD '04)*, Sydney, Australia, July 2004.
111. K. van den Doel, P. G. Kry, and D. K. Pai. Foleyautomatic: physically-based sound effects for interactive simulation and animation. In *Proc. 28th annual conf. on computer graphics and interactive techniques (SIGGRAPH '01)*, pages 537–544, New York, NY, USA, 2001. ACM.
112. K. van den Doel and D. Pai. Modal synthesis for vibrating objects. In K. Greenebaum and R. Barzel, editors, *Audio Anectodes*. A. K. Peter, Natick, MA, 2003.
113. M. van Walstijn. *Discrete-time modelling of brass and reed woodwind instruments with application to musical sound synthesis*. PhD thesis, Faculty of Music, University of Edinburgh, 2002.

114. N. J. Vanderveer. *Ecological acoustics: human perception of environmental sounds.* PhD thesis, Cornell University, 1979.
115. L. Verlet. Computer "Experiments" on Classical Fluids. I. Thermodynamical Properties of Lennard-Jones Molecules. *Physical Review*, 159(1):98+, July 1967.
116. Y. Visell, J. R. Cooperstock, B. L. Giordano, K. Franinovic, A. Law, S. McAdams, K. Jathal, and F. Fontana. A vibrotactile device for display of virtual ground materials in walking. In *Haptics: Perception, Devices and Scenarios*, number 5024 in Lecture Notes in Computer Science, pages 420–426. Springer, Berlin/Heidelberg, 2008.
117. Y. Visell, F. Fontana, B. Giordano, R. Nordahl, S. Serafin, and R. Bresin. Sound design and perception in walking interactions. *International Journal of Human-Computer Studies*, 67(11):947 – 959, 2009. Special issue on Sonic Interaction Design (SID).
118. L. Vu-Quoc and X. Zhang. An elastoplastic contact force-displacement model in the normal direction: displacement-driven version. *Proc.: Mathematical, Physical and Engineering Sciences*, pages 4013–4044, 1999.
119. W. H. Warren and R. R. Verbrugge. Auditory perception of breaking and bouncing events: a case study in ecological acoustics. *J. of Experimental Psychology: Human Perception and Performance*, 10(5):704–712, 1984.
120. H.-Y. Yao and V. Hayward. An experiment on length perception with a virtual rolling stone. In *Proc. EuroHaptics Int. Conf.*, pages 275–278, 2006.
121. P. Zahorik, D. Brungart, and A. Bronkhorst. Auditory distance perception in humans: A summary of past and present research. *Acta Acustica united with Acustica*, 91:409–420, 2005.
122. S. Zambon. Personal communication, 2007.
123. M. Zampini and C. Spence. The role of auditory cues in modulating the perceived crispness and staleness of potato chips. *Journal of Sensory Studies*, 19(15):347–363, 2004.